Netpump Data for Azure — Installation

Guide

*This guide provides a foolproof, step-by-step process to deploy **Netpump Data** on Azure. It is written for junior cloud administrators with limited Azure experience, so each step is ultra-explicit.*

Follow the steps in order and use a "build sheet" (e.g. a local text file or spreadsheet) to record important values (IDs, secrets, URLs) as you go. Browser-specific tips, common pitfalls, and validation checks are included to ensure a smooth setup.

Contents

Before You Start – Quick Checklist

Make sure you have the following prerequisites ready **before** proceeding. You can tick each item off as you confirm it:

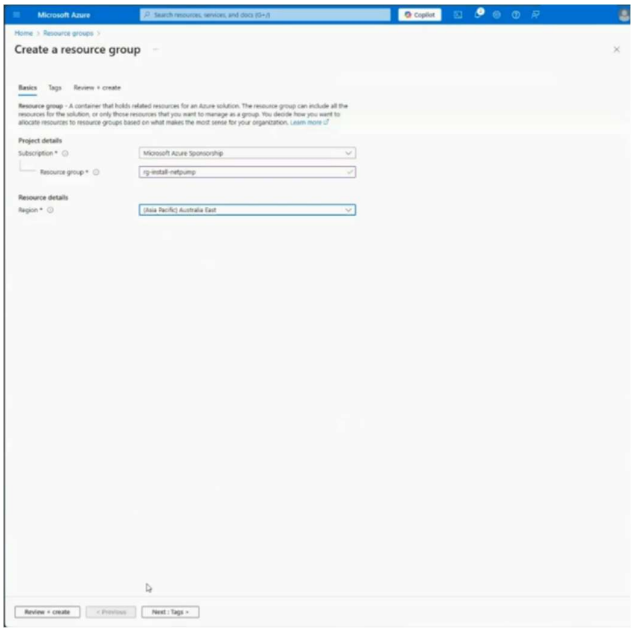
Item	Why it matters
Global Administrator rights in **Microsoft Entra ID** (Azure AD)	Needed to grant consent and assign app roles in Azure AD.
Owner role on the target **Azure subscription**	Needed to create resource groups and the managed application.
Decided on an Azure **region** (e.g. West Europe)	All resources must reside in the same region for this setup.
A naming **prefix** for resources (e.g. netpump-prod)	Keeps every Azure resource easily identifiable.
Your public **DNS zone** (e.g. pump.example.com)	Required for creating DNS CNAME records later.
A local **"build sheet"** (spreadsheet or text file) open	Where you will paste IDs, secrets, and URLs as you obtain them

Now, let's begin the installation step-by-step.

1. Create Two Resource Groups

1. **Create the core resource group:** In the Azure Portal, go to **Resource groups** and click **Create.** In the *Basics* tab of the **Create a resource group form,** enter the following:
   - **Subscription:** *Select your target subscription* (ensure you have Owner access on it).
   - **Resource group name:** rg-netpump-core (using your chosen prefix).
   - **Region:** *Your chosen region* (e.g. West Europe).

   Click **Review + Create,** then **Create.** Wait for the notification that says "Deployment succeeded." This confirms the resource group was created successfully **.**



Azure Portal – *Resource Group creation* form with the Name and Region filled in.

2. **Create the install resource group:** Repeat the above process to create a second resource group named `rg-netpump` install in the **same region**. This second group will be used to hold temporary installation assets separately from core resources. Again, wait for the *Deployment succeeded* message.

   Why two resource groups? Separating core resources and installation assets helps with organization and cleanup. The `rg-netpump-core` group will contain the persistent Netpump service resources, while `rg-netpump-install` will contain short-lived setup resources that can be isolated or removed later.

2. Add Key Vaults to the Resource Groups

You will create two Azure Key Vaults – one in each resource group:

1. **Core Key Vault:** In the **rg-netpump-core** resource group**,** click **+ Create** and search for **Key Vault.** Select **Key Vault** and click **Create.** In the Key Vault creation form:
   - **Key Vault Name**: `kv-netpump-core` (must be globally unique within Azure, usuallythe prefix plus core is fine).
   - **Region:** *Same region as above*.
   - Leave other settings at defaults (for this setup, standard sku and default access policy settings are fine).

   Click **Review + Create,** then **Create.** Wait for the Key Vault deployment to succeed.

2. **Install Key Vault:** Repeat the process in the **rg-netpump-install** group**.** Create a KeyVault named `kv-netpump-install` (same region). Wait for deployment success.

Why two Key Vaults? The core vault (kv-netpump-core) is intended to store long-lived secrets like cluster certificates, while the install vault (kv-netpump-install) will store short-lived secrets used during the installation process. This separation adds security and clarity.

Validation: After creation, verify in the portal that **each resource group contains one Key Vault** with the expected name. For example, in *rg-netpump-core* you should see kv-netpump-core, and in *rg-netpump-install* you should see kv-netpump-install.

3. Register the Server Application (Back-end AAD App)

Next, set up an **Azure AD app registration** to represent the Netpump server cluster (the back-end API):

1. In the Azure Portal, open **Microsoft Entra ID** (Azure Active Directory). From the left menu, select **App registrations**, then click **+ New registration**.

2. Fill out the *Register an application* form:
   - **Name:** NetpumpServerCluster (or a similar name identifying the Netpump servercluster).
   - **Supported account types:** Choose **Accounts in this organizational directory only** (Single tenant).
   - **Redirect URI:** Leave this blank for the server app (no redirect URI needed for a backend service).

   Click **Register** to create the application.

3. After registration, you will be taken to the **NetpumpServerCluster** app overview. **Copy the following IDs** from the overview and paste them into your build sheet for later use:
   - **Directory (tenant) ID** – (GUID identifying your Azure AD tenant).
   - **Application (client) ID** – (GUID for this NetpumpServerCluster app).

4. Still on the NetpumpServerCluster app, configure it to expose an API:
   - In the app's left-hand menu, go to **Expose an API.**
   - Click **Set** (or **Add an Application ID URI**). Accept the suggested default Application ID URI (it will look like api://<Application-ID-guid> or similar) or adjust it if needed (e.g**., api://netpump-prod-<GUID>**). Then click **Save.**
   - Now click **Add a scope.** In the **Add an API scope** form, set:
     - **Scope name:** Data.Transfer.All
     - **Admin consent display name:** Transfer data via Netpump
     - **Admin consent description:** *Allows transferring data through Netpump servers.*
     - **State:** Enabled (Yes).

Click **Add scope** to create this scope.

Validation: Back on the *Expose an API* page, you should now see the new scope listed, and its **Enabled** status should show "Yes". This confirms your API scope was created successfully.

4. Register the Client Application (Front-end AAD App)

Now create a second Azure AD app registration to serve as the client/front-end application (for the Netpump UI or user interactions):

1. In **App registrations,** click **+ New registration** again. Create another app with:
   - **Name:** NetpumpClient (to denote this is the client-facing app).
   - **Supported account types: Accounts in this organizational directory only** (Single tenant).
   - **Redirect URI:** Leave blank for now (we will configure authentication later).

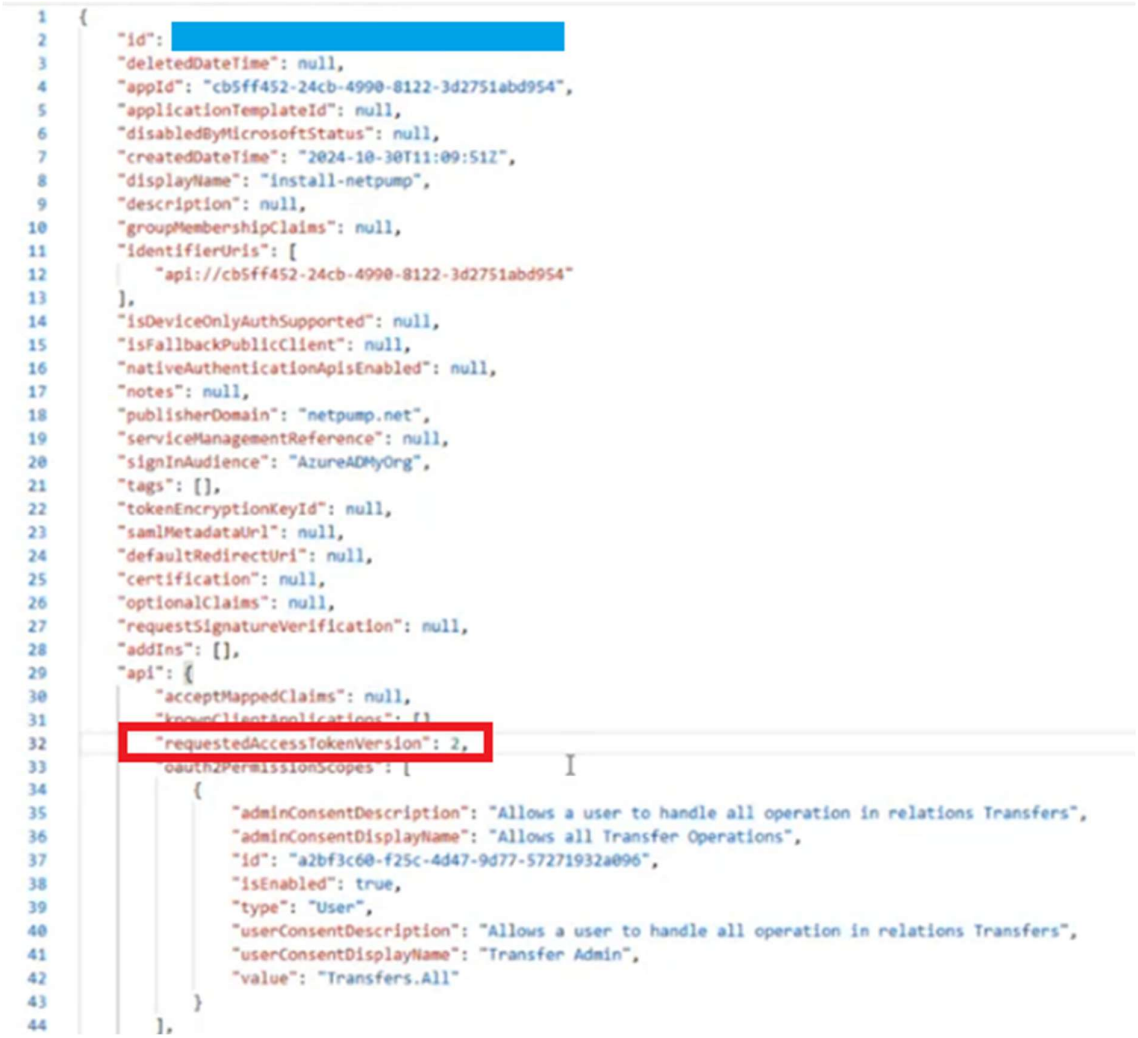
   Click **Register**. Once created, note the **Application (client) ID** for NetpumpClient (copy it to your build sheet; you already have the tenant ID from before, which is the same for both apps).

2. We need to define an **app role** in this client app, so that certain users (or the Netpump server itself) can be assigned as administrators:
   - In the NetpumpClient app, open the **Manifest** (you'll find "Manifest" in the left menu of the app registration blade).
   - The manifest is a JSON file. Find the "appRoles": [] section. Insert a new object inside the array to define a role. For example, if the appRoles section is empty ("appRoles": []), you can replace it with the following (make sure it's inside the square brackets and comma-separated if there are other roles):

json

```
{
    "allowedMemberTypes": [ "User", "Application" ],
    "description": "Administers Netpump servers",
    "displayName": "ServerAdmin",
    "id": "<NEW-GUID>",
    "isEnabled": true,
    "value": "ServerAdmin"
}
```

You will also need to verify "requestedAccessTokenversion": 2 . If it appears with a NULL value, change the value to "2" as shown.

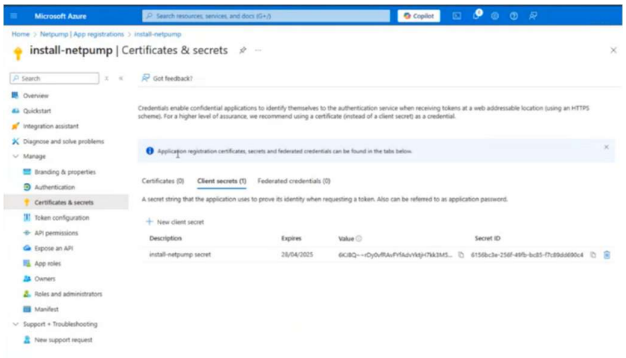


- **Important:** Replace `<NEW-GUID>` with a new unique GUID. (You can generate a GUID using a tool or online GUID generator. Every app role `id` must be a unique GUID.)
- Save the manifest changes. *(The portal will validate the JSON – ensure the syntax is correct.)*

3. Next, grant this client app permission to call the server app's API:
   - In the `NetpumpClient` app, go to **API Permissions.**
   - Click **+ Add a permission** → select **My APIs** → you should see `NetpumpServerCluster` in the list. Select it.
   - Under the **Delegated permissions** section, find and **check** the `Data.Transfer.All` scope (the one you created in the server app) and then click **Add permissions**.
   - The permission will be added with a warning that **admin consent** is required. Click the **Grant admin consent for [Your Tenant]** button. Confirm the consent when prompted. This grants tenant-wide consent for your client app to call the server app's API (so users won't need to consent individually).

Validation: After granting consent, the API permission should show the status **Granted for [Your Tenant]**. This indicates the permission is active. If you refresh the API Permissions page, you shouldn't see a "Not granted" warning anymore.

4. Lastly for the client app, create a **client secret** that the Netpump service will use to authenticate as this app:
   - In the NetpumpClient app, go to **Certificates & secrets.**
   - Click **+ New client secret.** For **Description,** enter something like svc-auth (to indicate this secret is for the service authentication). Set an appropriate expiration (e.g., 24 months).
   - Click **Add**. Azure will generate a new secret**. Copy the secret's Value** immediately and paste it into your build sheet (e.g., under "Client Secret"). **You will not be able to view this value again after you leave the page.**



Client Secret creation panel with the **secret value visible** (highlight the Value field in the screenshot). This is to show how the secret appears once generated.

Common Pitfall: Not copying the secret value right away. If you navigate away, the value will be hidden forever and you'd have to create a new secret. Always store it securely as soon as it's created.

5. Configure Authentication for the Client App

Now that the two app registrations are created, configure the authentication settings for the client app (NetpumpClient) so that it can be used for user sign-ins (including from the desktop GUI, if used):

1. In the NetpumpClient app registration, go to Authentication in the menu.

2. Under **Platform configurations**, click **+ Add a platform** and choose **Web** (since the Netpump server will interact with this as a web client).
   - Add a **Redirect URI** of type Web: enter https://localhost (we use localhost as a placeholder redirect URI for testing/desktop app scenarios).

- Under **Implicit grant and Hybrid flows**, check the boxes for **Access tokens** and **ID tokens**. (This enables OAuth2 implicit flow if needed and ensures an access token and ID token are provided in the response.)
- Click **Configure** to save this platform configuration.

3. *(Optional)* If you anticipate building a single-page application or JavaScript front-end running on a different port (for example, a development React app on http://localhost:3000), you can also add a **SPA** platform with that redirect URI. This step is not necessary for the basic installation, but can be done similarly by selecting **SPA** and adding http://localhost:3000 as a redirect URI (and enabling tokens).

4. Click **Save** on the Authentication page if it didn't auto-save. The client app is now configured to allow interactive user sign-ins.

   Validation: After saving, you should see the new platform (Web) listed with the URI, and the checkboxes for Access tokens/ID tokens should remain checked. This confirms the client app is ready for user authentication flows.

6. Assign the Admin Role to Yourself (or a Group)

Recall we created an app role ServerAdmin in the NetpumpClient app's manifest. Now we must assign this role to the appropriate user(s) who will administer the Netpump servers (likely you, and/or a group of admins):

1. In the Azure Portal, go to **Microsoft Entra ID** (Azure AD) and navigate to **Enterprise applications** (this is the list of service principals for your apps). Locate the application **NetpumpClient** in the list (you can use the search bar if needed) and click it.

2. In the NetpumpClient enterprise application blade, go to **Users and groups** in the left menu.

3. Click **+ Add user/group** at the top. In the Add Assignment pane:
   - Click **Users > None** Selected to open the user picker. Find and select your own user account (or an Azure AD group containing the relevant administrators). Click **Select.**
   - Now click **Roles > None Selected**. You should see the **ServerAdmin** role we created. Select **ServerAdmin** and click **Select**.
   - Click Assign to assign the selected user (you) to the ServerAdmin role for the NetpumpClient app.

4. After assignment, you can verify it's in place: the user or group should now be listed under Users and groups with **Role** showing as **ServerAdmin.**

   Validation: Refresh the Users and groups page for NetpumpClient. You should see your account (or chosen group) listed with **ServerAdmin** under the Role column.

This means you (or your admins group) now have the elevated role needed to manage Netpump servers via the application.

Why do this? Assigning the ServerAdmin role ensures that only authorized users can administer the Netpump Data service. In later steps, when you log into the Netpump interface, the system will check that your account has this role before granting admin access.

7. Grant Key Vault Access to the Client App

The Netpump deployment (running under the context of the NetpumpClient app) will need to retrieve secrets (like certificates) from Azure Key Vault. We must grant it the necessary permissions on the Key Vaults:

1. Go to the **Key Vaults** blade and open the **kv-netpump-install** vault (the one in the install resource group).

2. In the key vault, under **Settings**, click **Access policies** (ensure you are using Vault access policy model, which is default).

3. Click **+ Create** (or **Add Access Policy**). In the Add access policy form:
   - For **Configure from template (optional)**, you can leave it as "None" (we'll select permissions manually).
   - Under **Key permissions** you can leave unselected (no key needed for now).
   - Under **Secret permissions**, select **Get** and **List**.
   - Under **Certificate permissions**, select **Get** and **List** as well. (This will allow the app to retrieve certificate secrets from the vault.)
   - In the **Principal** field, click **None selected** to open the principal picker. Search for **NetpumpClient** (the name of the app registration we created). You should see it appear (it will have a guid ID associated). Select it and click **Select**.
   - Click **Add** to add this access policy, then **Save** on the Access Policies page to apply it.

4. (Optional) If you anticipate that the Netpump service will need to access the core vault (kv-netpump-core) at runtime (for example, to fetch cluster certificates stored there),you can repeat the above steps for kv-netpump-core as well. Grant the same **Get** and **List** for secrets and certificates to the **NetpumpClient** principal on that vault. (This step can be done later or as needed; it's not strictly required for the initial deployment, since we will put our certificate in the install vault.)

   Validation: After saving the access policy, in the **Access policies** list for kv-netpump-install, you should see an entry for **Principal: NetpumpClient** with the selected **Secret Permissions** and **Certificate Permissions** listed (Get, List). This indicates the client app (and thus our Netpump service) can read secrets from that vault. If you open the kv-netpump-install **Overview** and click **Managed**

identities, you will *not* see the app here because we added an access policy directly for the service principal; that's expected.

Common Pitfall: Selecting the wrong principal or vault. Ensure you picked the **NetpumpClient** app (not the server app) for the access policy, and that you applied it on the **kv-netpump-install** vault. A simple mistake here could cause permission issues when the application tries to retrieve secrets later.

8. Generate the SSL Certificate in Key Vault

Netpump requires an SSL certificate for secure communication. We will generate a self-signed certificate in the Key Vault to use for this purpose:

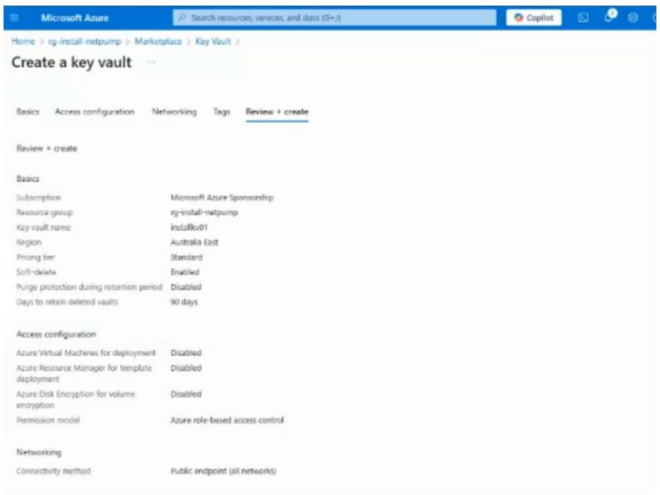
1. Open the **kv-netpump-install** Key Vault (the install vault). In the vault, go to the **Certificates** section and click **+ Generate/Import**.

2. In the **Create a certificate** screen:
   - **Method of Certificate Creation:** Choose **Generate** (to create a new certificate).
   - **Certificate Name**: netpump-tls (you can use this name for clarity; it will be the friendly name of the cert).
   - **Type of Certificate Authority (CA)**: Select **Self-signed** (i.e., generate a self-signed certificate). *(In some Azure Portal versions, you might directly have an option like "Self-signed certificate" to choose.)*
   - You can leave other fields at defaults (Key Type RSA, Key Size 2048, etc., which are fine for a self-signed TLS cert).

   Click **Create** to generate the certificate. The vault will create a new certificate and also generate an associated secret (with the same name) containing the certificate's private key.

3. It may take a minute for the certificate to be created. Once the **Certificate Operation** shows as completed (Status: Issued), click on the new certificate (netpump-tls) in the Certificates list.

4. In the certificate's detail page, find the **Secret Identifier** (a URL). This is typically labelled as **Secret Identifier** and looks like https://kv-netpump-install.vault.azure.net/secrets/netpump-tls/<GUID>. Copy this **Secret Identifier URL** and paste it into your build sheet (label it "Cert URL").



Key Vault Certificate details showing the Secret Identifier URL for the `netpump-tls` certificate (highlight the URL). This will be used by the Netpump service to fetch the certificate at runtime.

> **Validation:** The Secret Identifier URL should start with your key vault's URI and include `netpump-tls` and a version GUID. If you accidentally copy the **Certificate Identifier** or something else, the deployment will fail. Double-check you have the **Secret Identifier** (which typically differs by the segment `/secrets/` in the URL). **Common Pitfall**: Copying the wrong URL. Ensure it's the secret URL (for the private key), not just the certificate (public key) URL.

9. Create a Storage Account and an SMB File Share

Netpump uses an Azure Storage File Share (SMB share) as part of its data transfer pipeline (for staging data, etc.). We will create a storage account and a file share:

1. In Azure Portal, click **+ Create a resource** and search for **Storage account**. Select **Storage account** and click **Create**.

2. In the *Create a storage account* form (Basics tab):
   - **Subscription**: *Your subscription*.
   - **Resource group**: Select `rg-netpump-core` (the core resource group).
   - **Storage account name**: `stnetpumpdata` (as an example - storage account names must be all lowercase, 3-24 characters, and globally unique. You can use your prefix, e.g., `st<prefix>data`. If `stnetpumpdata` is taken, choose a different variation).
   - **Region**: (should auto-fill with the resource group's region, ensure it's the same region as everything else).
   - **Performance/Redundancy**: You can leave the defaults (Standard performance, Locally-redundant storage (LRS) is fine unless your scenario needs geo-redundancy).

- Leave other settings at defaults unless you have specific requirements.

Click **Review + Create**, then **Create**. Wait for the deployment to finish (it might take a minute for the storage account to be ready).

3. Once the storage account is created, navigate to the storage account resource(stnetpumpdata). In the left menu under **Data storage**, click **File shares**.

4. Click **+ File share**. Create a new file share:
   - **Name**: data (this will be the share name; you can choose another name but "data" is simple for our guide).
   - **Quota**: leave default or set as needed (the default is fine for now).

   Click **Create** to create the file share.

5. Now retrieve the connection string (or at least the key) for the storage account:
   - In the storage account's menu, go to **Access keys**.
   - You will see two keys (key1 and key2). Click **Show** to reveal **Key1**, and copy the Connection string for Key1. Paste this connection string into your build sheet (label it "Storage connection string"). The connection string contains the storage account name and the key.

   *(Alternatively, you can copy the storage **account name** and **key1** value separately into the build sheet. The upcoming deployment will ask for these pieces. The connection string just conveniently packages them, but we will likely input them separately.)*

   Validation: Ensure the file share was created by checking the **File shares** list for an entry named data. Also verify you have the correct storage credentials: the account name and one of its access keys. A quick way is to verify the connection string you copied starts with DefaultEndpointsProtocol=https;AccountName=stnetpumpdata;... and contains AccountKey=... with a long base64 string — that confirms you have the name and key captured.

   Common Pitfall: Forgetting to create the file share, or copying only the key but not the name. The Netpump service needs both the share name and the key (or connection string) to connect. Make sure you have the share name (data), the storage account name, and the key in your notes.

10. Deploy Netpump Data from the Azure Marketplace

With all prerequisites in place (app registrations, Key Vault, certificate, storage, etc.), you can now deploy the Netpump Data service via the Azure Marketplace offering:

1. In Azure Portal, click **+ Create a resource** and search for **Netpump** in the Marketplace. Locate **Netpump Data** (by Pacbyte) and select it. Click **Create** to start the deployment wizard for this managed application.

2. Basics: On the first screen, provide the basic settings:
   - **Subscription**: *Your subscription*.
   - **Resource Group**: Select rg-netpump-core (the core group where the managed app will reside).
   - **Region**: Choose the same region you've been using (all resources need to align here).
   - **Application Name**: netpump-prod (for example – this will name the Netpump application instance. You can use your prefix or a descriptive name).
   - You might see other basic fields like Project details or instance details; fill them as appropriate, but the above are key.

3. **Plan**: On the next screen, choose the appropriate plan/SKU for Netpump Data.

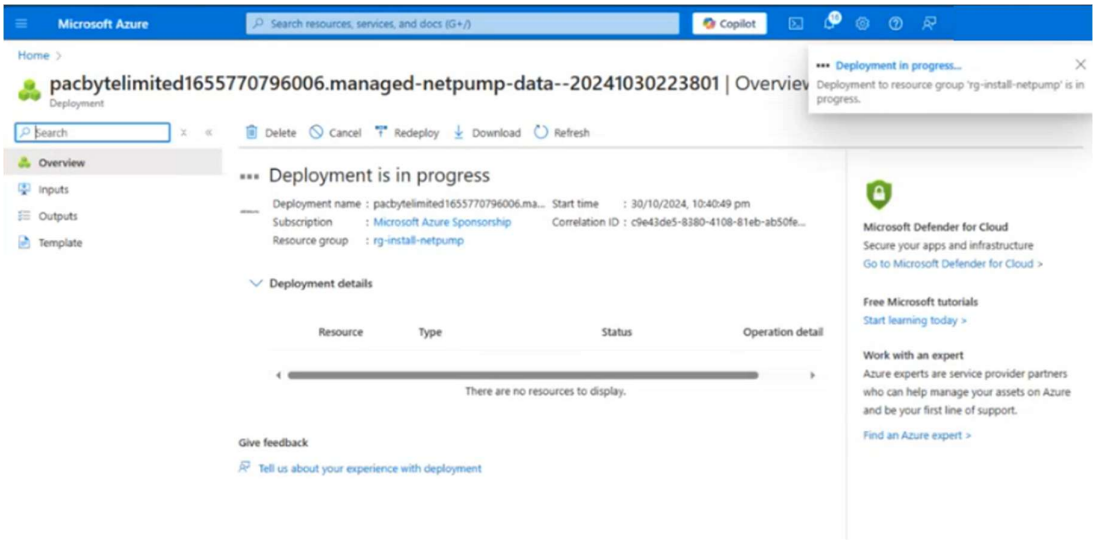
4. **Parameters**: This section will have input fields to configure the Netpump deployment. Here is where you paste the values from your build sheet:
   - **Tenant ID** – Paste the Azure AD Directory (tenant) ID (GUID).
   - **Client ID** – Paste the Application (client) ID of the NetpumpClient app (the front-end app).
   - **Client Secret** – Paste the client secret value that you saved for NetpumpClient.
   - **Certificate URL** – Paste the Key Vault Secret Identifier URL for the netpump-tls certificate (from step 8).
   - **Storage Account Name** – e.g., stnetpumpdata (the name of the storage account you created).
   - **Storage Key or Connection String** – depending on the field: if there is a field for the key, paste the storage account Key1 value; if it asks for a connection string, paste the connection string for the storage account.
   - **Static IP Allowlist (optional)** – If the deployment form includes an option to specify allowed IP ranges (for accessing the Netpump service), you can enter your corporate IP or range here. If you leave it blank, it will allow default/open access (or whatever the default in the template is).
   - Any other parameters required.

   Double-check each value carefully. **Typos in these values are the #1 cause of deployment failures.** Common things to verify:

   - No extra spaces or missing characters in the GUIDs and keys.
   - The correct secret (ensure it's the one for the client app, not some other secret).
   - The correct URL (should be the one you copied with /secrets/netpump-tls/).

5. Once all parameters are filled, click **Review + Create**. The portal will validate the inputs. If validation passes, click **Create** to begin deployment of the Netpump managed application. This will kick off the creation of all necessary Azure resources (VMs, networking, etc.) for Netpump.

6. Wait for the deployment to complete. This can take several minutes as it provisions VM instances, configures networking, etc. Keep an eye on the notifications for "Deployment succeeded."



Azure Portal – *Deployment in progress* confirmation page for the Netpump Data deployment.

Validation: If the deployment finishes with a status **Succeeded**, all Netpump resources have been created. If it **fails**, note the error message and consult it. The most common deployment failures at this stage are due to incorrect values in the parameters (for example, an incorrect **Client Secret**, a wrong **Certificate URL**, or a typo in the **Storage account name/key**). If you get a failure:

- Go back and **double-check the values** in your build sheet and in the form, fix any issues, and try the deployment again.
- You can find detailed error logs by clicking on the deployment in the **Notifications** or in the resource group's **Deployments section**.

11. Retrieve Deployment Outputs and Configure DNS Records

Once the Netpump managed application is deployed, it will provide output values (such as the fully qualified domain names of the Netpump server VMs). We will use these to set up friendly DNS names:

1. In the Azure Portal, navigate to the resource group `rg-netpump-core` where you deployed Netpump. You should see a resource of type **Managed Application** (or it might appear as a group of resources labelled with the application name you gave, e.g.,`netpump-prod`).

2. Click on the Netpump managed application resource (it might have the same name you provided as "Application Name"). In its overview or left-hand menu, find Outputs. (If it's a classic ARM template deployment, you might instead go to the Deployments section and check the output there.)

3. In the Outputs, identify the fully qualified domain names (FQDNs) for the Netpump server instances. For example, you might see outputs like:
   - `pump0_FQDN` : `netpump-0.westeurope.cloudapp.azure.com`
   - `pump1_FQDN` : `netpump-1.westeurope.cloudapp.azure.com`
     (Your naming might differ, but essentially there will be one FQDN for each VM in the cluster.) Copy these FQDNs to your build sheet or make note of them.

4. Now, configure DNS **CNAME** records in your public DNS zone (the one you identified earlier, e.g., `pump.example.com`) to map friendly names to these Azure FQDNs:
   - For the first server: create a CNAME record with **Name** `pump1` (for example) and Value set to the first FQDN (`netpump-0.westeurope.cloudapp.azure.com` or as appropriate).
   - For the second server: create a CNAME with **Name** `pump2` pointing to the second FQDN.

   This will allow users to access the servers via e.g. `pump1.your-domain.com` and `pump2.your-domain.com`. (If your DNS zone is exactly `pump.example.com`, then `pump1.pump.example.com` will be the full address.)

   You can use the DNS management interface of your DNS provider or Azure DNS zone to add these records. In Azure DNS, you would go to your DNS zone resource and use ***+ Record*** *set to add CNAMEs.*

Validation: After adding the records, test name resolution. Open a command prompt or terminal and `ping pump1.your-domain.com` (replace with your actual domain). It should resolve to an Azure IP (you may not get ping replies if ICMP is blocked, but the resolution should show

the cloudapp.azure.com address). This confirms your DNS is set up. Keep in mind DNS propagation might take a few minutes depending on your TTL.

Note: We use CNAMEs so that the Azure-provided domain (which points to the VM's IP) is aliased behind your friendly name. If your organization manages DNS differently, ensure the Netpump servers are reachable via some DNS names that you can use internally.

12. Configure Each Netpump Server via the Admin UI

At this stage, the infrastructure is up, but each Netpump server instance needs to be configured with the settings (the IDs, secrets, and resource information you prepared). Netpump provides a configuration web interface on each server for initial setup:

1. Open a web browser and navigate to the configuration page of the first server: `https://pump1.<your-domain>/config` (for example, `https://pump1.pump.example.com/config`). Because we are using a self-signed certificate (generated in Key Vault), your browser will likely show a security/Certificate Warning (the certificate won't be trusted by your browser). **Proceed/ignore the certificate warning** to continue to the site:
   - In Chrome/Edge, you may need to click **Advanced** and then **Continue to site**.
   - In Firefox, you might add an exception.
   - *If the page fails to load, double-check your DNS is resolving correctly, and that the URL is* `https://` *(SSL) and not* `http`. *The Netpump UI likely requires HTTPS.*

2. You should see a **Netpump Configuration** page (a form) on pump1. Now, fill in **all fields** on the form using the values from your build sheet:
   - **Tenant ID**: The Azure AD Directory (Tenant) ID (GUID).
   - **Client ID**: The Application (Client) ID of your NetpumpClient app registration.
   - **Client Secret**: The client secret value you created for NetpumpClient.
   - **Certificate URL**: The Key Vault Secret Identifier URL for netpump-tls (the one you copied in step 8).
   - **SMB Share Path**: `\\<storage-account-name>.file.core.windows.net\<share>` – in our example, `\\stnetpumpdata.file.core.windows.net\data`. (This is the UNC path to the Azure file share you created.)
   - **SMB Username**: Azure\<storage-account-name> – for example, `Azure\stnetpumpdata`. (When connecting to an Azure Files share via SMB, the username is the word "Azure" followed by a backslash and the storage account name.)
   - **SMB Password**: The Storage account key (Key1 that you copied). This serves as the password for the SMB share.

   Double-check that each field is filled correctly with no typos. This is essentially the same info you provided during deployment, but now directly in the app's config.

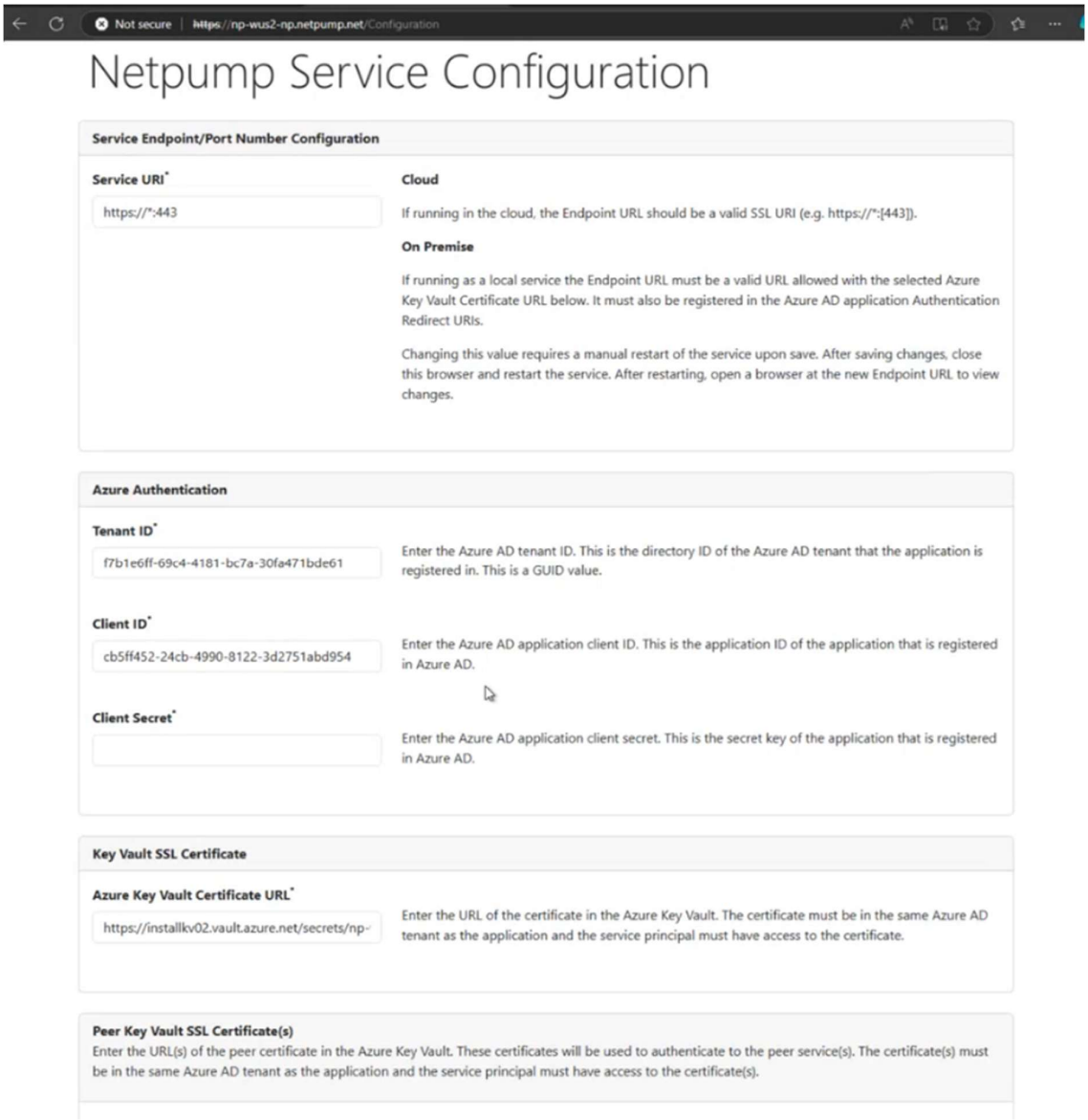
3. Click **Save Configuration** on the configuration page.
    - Once you click Save, the Netpump server will typically attempt to verify these settings and might initiate an authentication flow.
    - You should be prompted with a Microsoft Entra ID si**gn-in window** (this could be a pop-up or redirect for OAuth consent). **Log in with a Global Administrator account** for your Azure AD (the same one that has permission to consent to apps).
    - After signing in, you will see a permission request (because the Netpump server is asking to use the NetpumpClient app's credentials to access the API or directory). The prompt will likely say something about granting permissions (for example, it might list the Data. Transfer. All scope or other permissions the service needs).

    - Click **Accept** to grant consent. This is a one-time consent needed to allow the Netpump server to operate with the provided credentials.

4. After accepting, the page should return to the Netpump UI. Look for a confirmation message at the top of the config page. Ideally, you should see a green banner or text stating **"Settings updated successfully"** (or similar positive confirmation).
    - If there is an error message instead, re-check the values you entered. A common mistake is an incorrect secret or a mis-typed GUID.
    - If the page seems to hang after login, ensure that the pop-up was not blocked by your browser. Also ensure you used a Global Admin as standard users might not be allowed to consent depending on your tenant settings.

5. Now repeat the configuration for the second server:
    - Visit https://pump2.<your-domain>/config (e.g., https://pump2.pump.example.com/config).
    - Again, ignore the certificate warning and fill in the exact same values (Tenant ID, Client ID, Secret, Cert URL, SMB path, username, password).
    - Save and again sign in with Global Admin to accept permissions on this server as well. You should get the same "success" indication for pump2.

    Each Netpump server in the cluster needs to be configured. If you had more nodes, you'd repeat for each. In our two-node example, you've done both.



Netpump server *Configuration*

Validation: To confirm that each Netpump server is properly configured, you can navigate to https://pump1.<your-domain>/status (and similarly pump2). This should display a JSON status, for example:

json

{ "ready": true }

This indicates the server is up and running with the configuration. (You might do this after the next step as well.)

Common Pitfalls:

- Not using a Global Admin for the consent step (resulting in an “insufficient privileges” error).
- Browser blocking the sign-in pop-up (ensure pop-ups are allowed for your Netpump site or use a new tab via copy-paste if needed).
- Typos in the configuration values (which could lead to an error or the service not being able to connect to Azure resources).
- If you encounter issues, you can revisit the config page by navigating to the URL again; it will show the fields (possibly pre-filled if it saved some) and errors if any.

13. (Optional) Install the Netpump Desktop GUI for End Users

Netpump also provides a Desktop client application for end-users to easily transfer data. If you wish to use/test the end-user experience, you can install this GUI client:

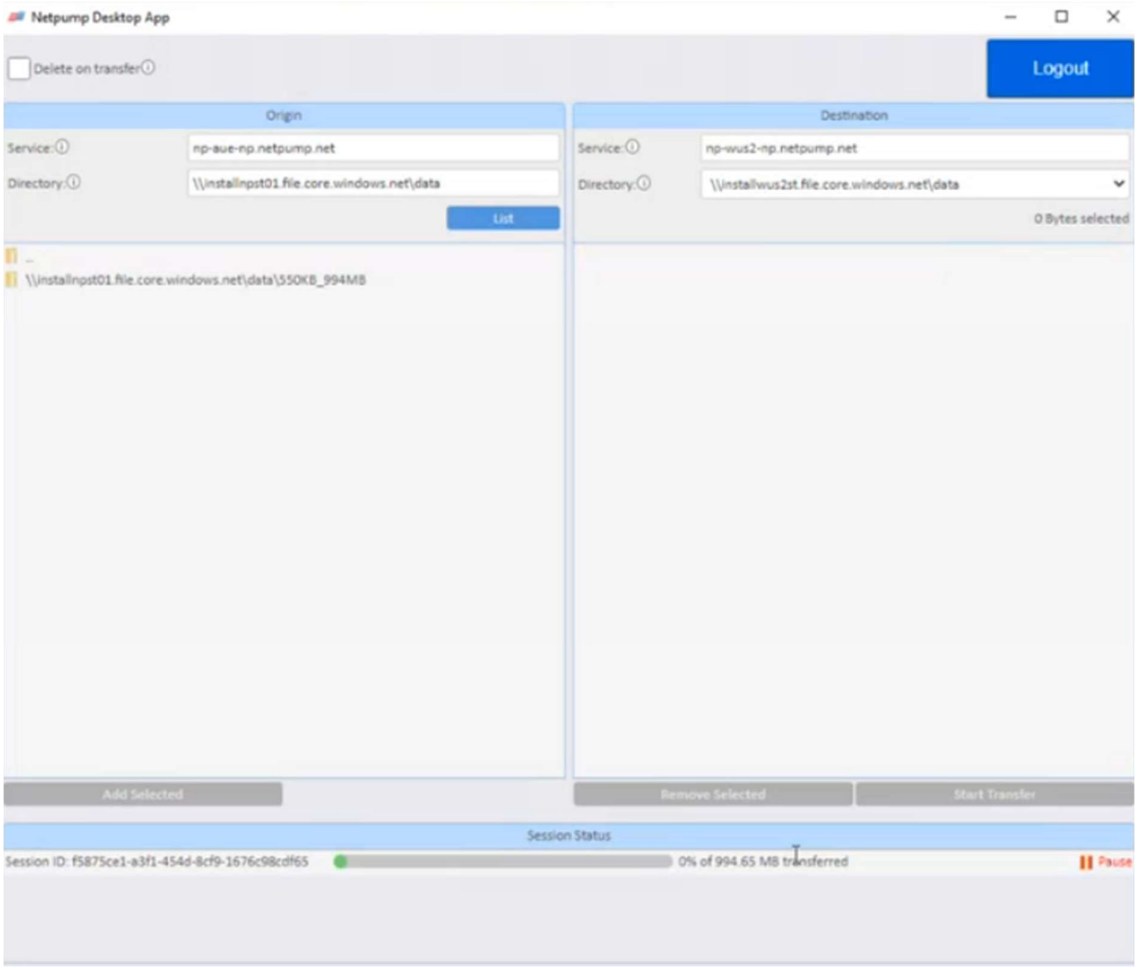
1. **Download the Netpump Desktop installer:** Click here to download the Windows Desktop installer.

2. Run the installer on your machine and follow the prompts to install Netpump Desktop.

3. Launch the **Netpump Desktop** application. On first launch, it will prompt you to **Sign in**. Use your Azure AD credentials (the user account that was given the Server Admin role, or any user authorized to use Netpump if applicable). This should trigger a login via your web browser or a pop-up – log in and grant any requested permissions.

4. Once signed in, you should see the Netpump desktop interface. You can now test transferring data:
   - Select an **Origin** service and a **Destination** service from the options (for example, one could be an on-prem folder or an Azure storage, and the other the Netpump service – refer to Netpump docs for how it appears, but likely the Netpump servers show up as a transfer target).
   - Choose some files or folders to transfer (the UI might allow you to browse and tick folders).
   - Click **Start Transfer**.

5. Monitor the transfer progress. You should see a progress bar or status indicator. When it's complete, it should show 100% and indicate success for the transfer.



Netpump Desktop application showing a transfer. This demonstrates that data transfer is working via the Netpump system.

This step is optional, but it's a good end-to-end test that the whole setup (client app, authentication, Netpump servers, and storage) is functioning for real file transfers.

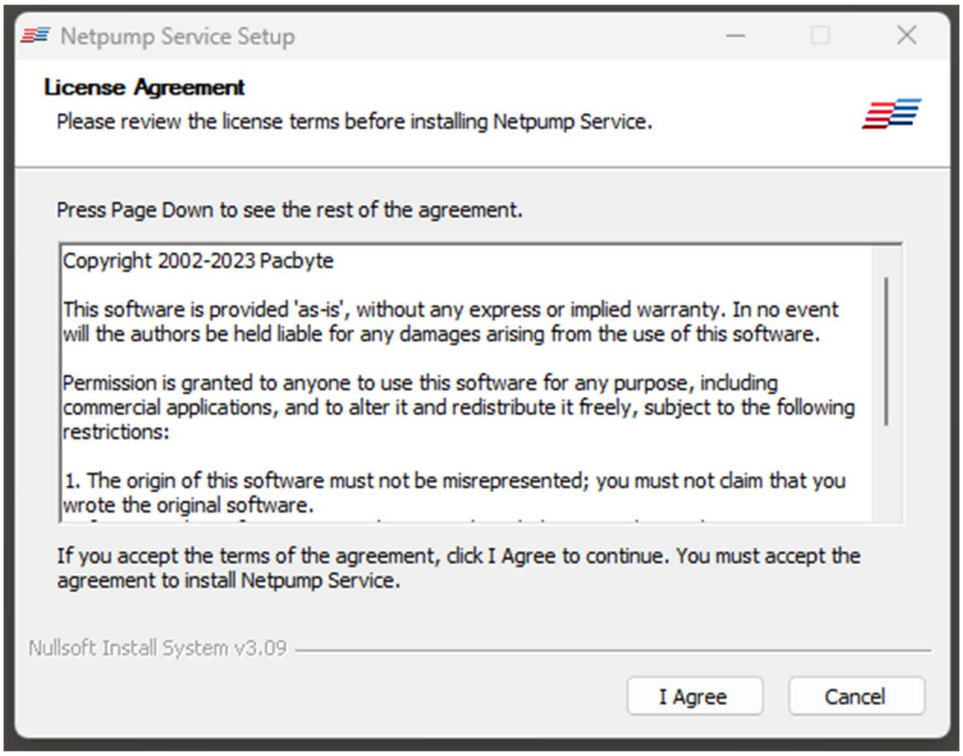
Validation: If the desktop client successfully transfers data and completes with no errors, it means the Netpump service is operating as expected. You could compare the transfer speed to a normal SMB copy to see the improvement (Netpump boasts significantly faster transfer, e.g., "20× faster than baseline SMB copy"). Also, check that the files indeed arrived at the destination (e.g., in the Azure Files share) intact.

14. Install Netpump Data - On Premises version for Internal Servers

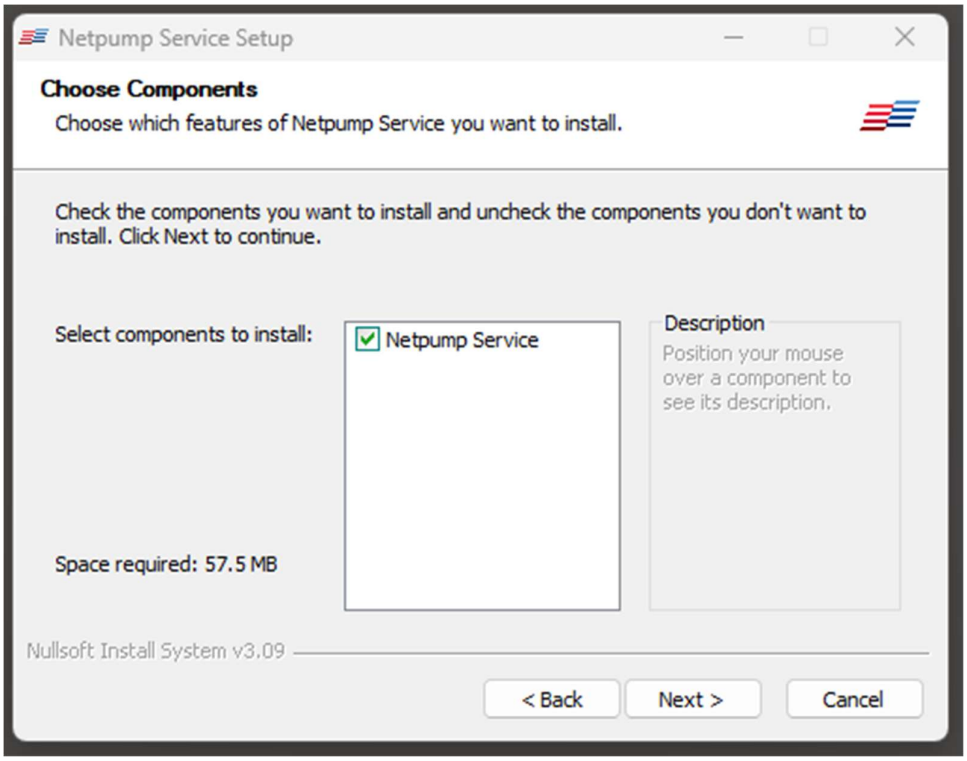
Download the Netpump On Premises installer: Click here to download the Windows On Premises installer.

Installing the Application

1. Open the downloaded installer file on your local machine.
2. Confirm the displayed End User License Agreement. This can also be viewed online at Netpump EULA.

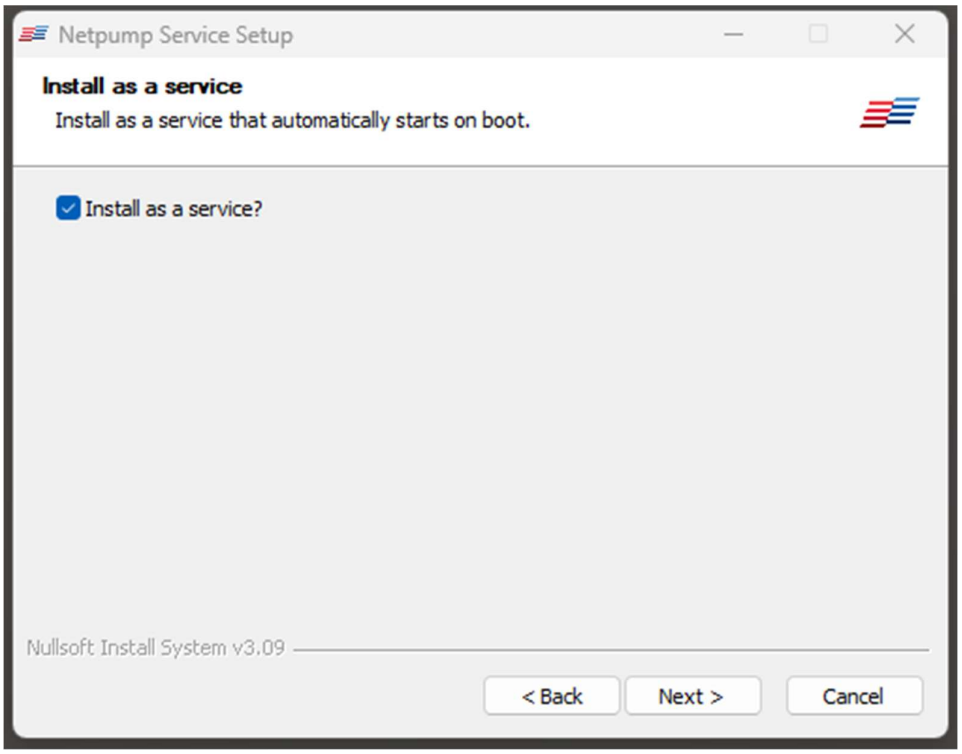


3. Select components to install.

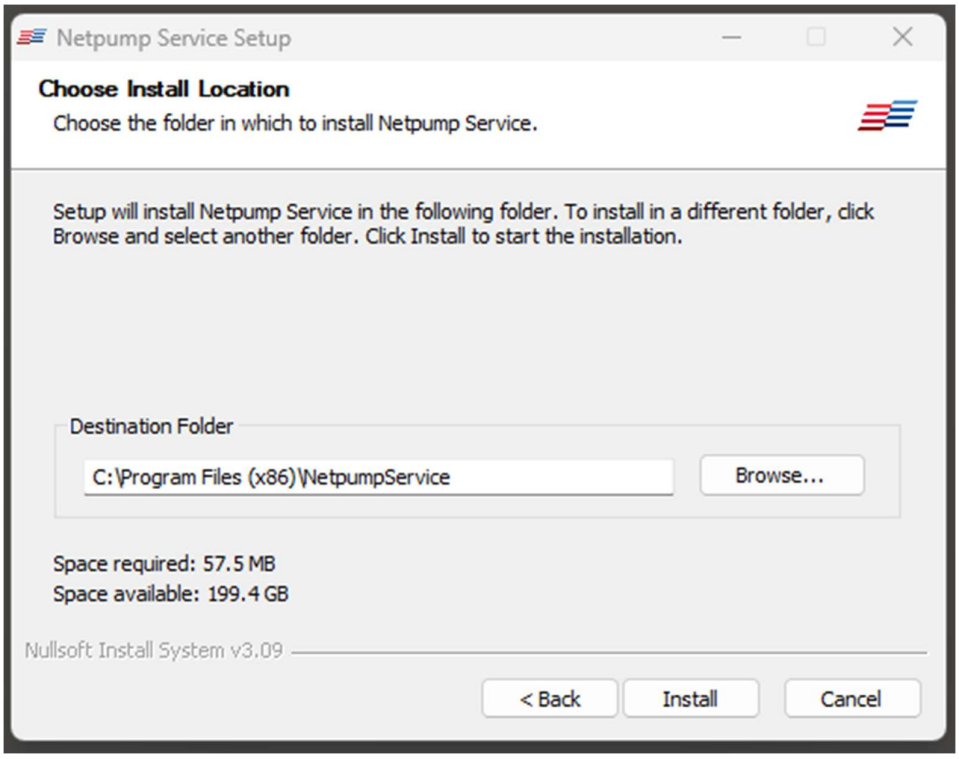


4. Select to Install as a service.

Select this option to have Netpump Service created as a Windows Service that starts automatically with machine boot.

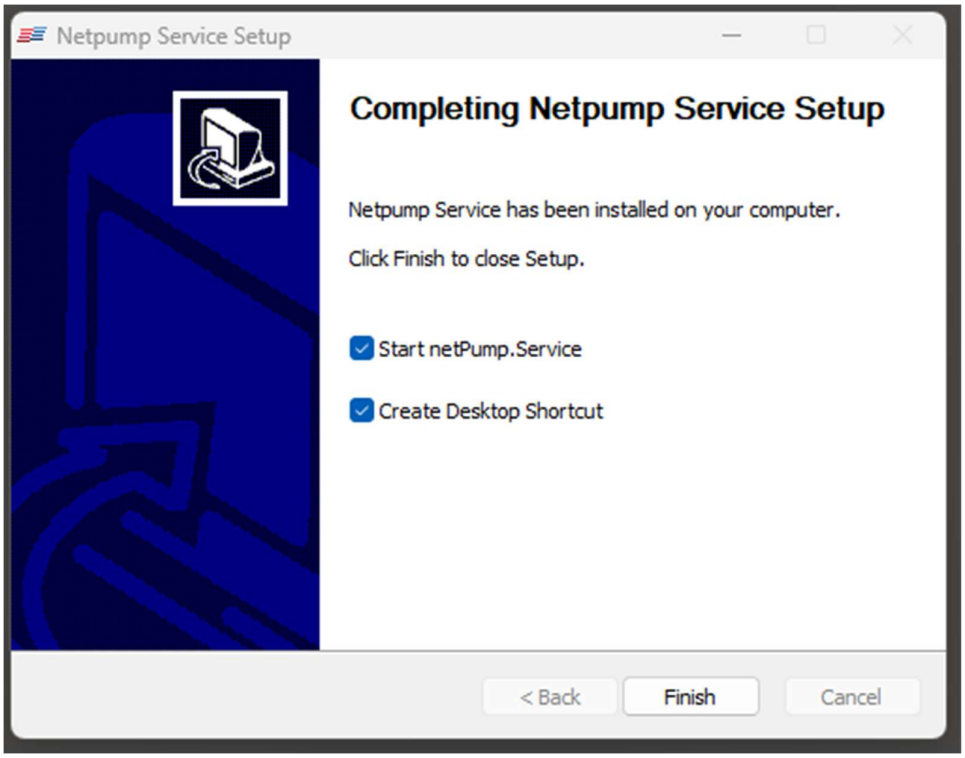


5. Select the installation directory.



6. Installation is now complete.

You may choose to start the service at this time and/or create a Desktop Shortcut.



Starting the Application

If created as a service the Netpump Service will start automatically on machine boot. To start/stop manually load Service and search for NetpumpService.

If not created as a service the Netpump Service can be started by clicking the Desktop Shortcut or executable 'netPump.Service.exe' file in the install directory.

Accessing the Netpump Service Administrator Page

Once the service has started, double click the created Desktop Short or executable file 'netPump.Service,exe' in the install directory. This will activate the Netpump Service Administrator Page. See section 12. Configure Each Netpump Server via the Admin UI above for configuration instructions.

15. Use of PowerShell for Scripting Functions

Netpump file transfers can be initiated programmatically over an HTTPS REST API. We supply a PowerShell module that provides a convenient interface to make these API calls using the Start-NetpumpTransfer command.

Authentication

The Netpump PowerShell script depends on Microsoft's Az PowerShell module.

Your PowerShell environment must call Connect-AzAccount to authenticate prior to calling Start-NetpumpTransfer.

The Azure account you use must have the Automation role for the Netpump Server App Registration in Azure.

Netpump module

The module is contained in Netpump.zip. Extract this zip file and use the PowerShell command Import-Module .\Netpump.psm1 to import it.

The Start-NetpumpTransfer command starts a transfer.

Run Get-Help Start-NetpumpTransfer to see further documentation.

Walkthrough: Azure Automation Runbook

In this example we will create an Azure Automation Runbook to initiate a Netpump file transfer.

We will create the Automation Account, grant the Automation Account's Service Principal permission to use the Netpump cluster, create a simple Runbook, and transfer a file.

Prerequisites

You will need two Netpump servers in a Netpump cluster already set up.

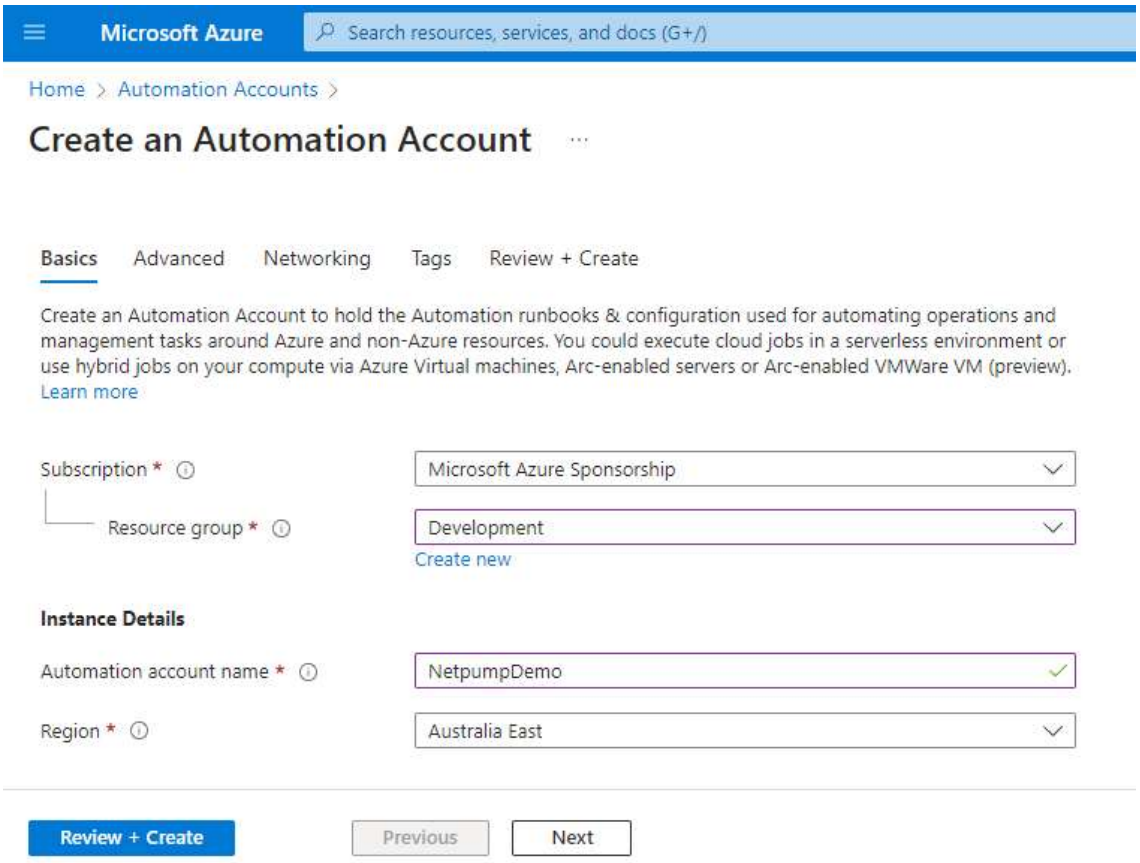
For the initial setup, you will need PowerShell with the Microsoft.Graph module installed. (Run Install-Module Microsoft.Graph -Scope CurrentUser, or see Microsoft.Graph documentation for further installation options.)

You will need to know the following information, which you would have used when setting up your Netpump servers:

- Tenant ID of your Azure Active Directory tenant.
- Object ID of your Netpump cluster Enterprise Application (note: the Enterprise Application is different to the App Registration).
- App Role ID of the Automation role you created during the App Registration
- URLs of the origin and destination Netpump servers
- Source UNC path of the file to transfer from the origin server
- Destination UNC path of where to store the file on the destination server

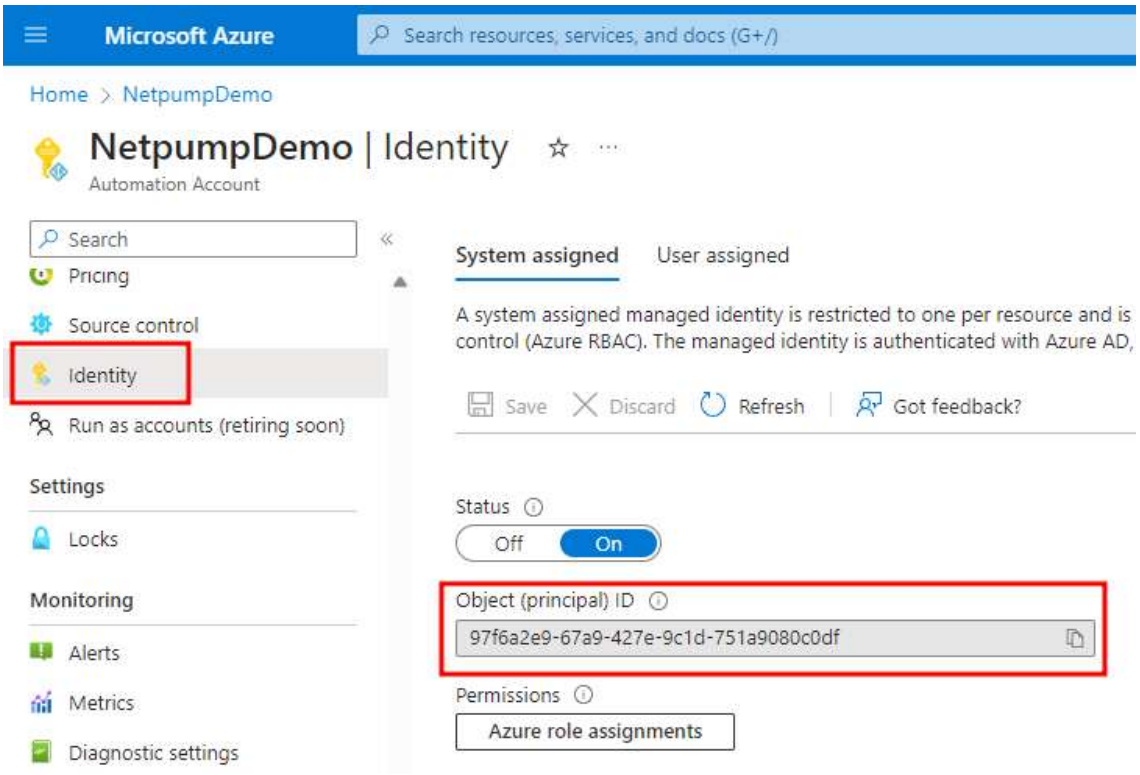
Steps

1. Download Netpump.zip which contains the PowerShell module. You will upload this to the Azure Automation Account later in this walkthrough.
2. In Azure Portal, create a new Automation Account

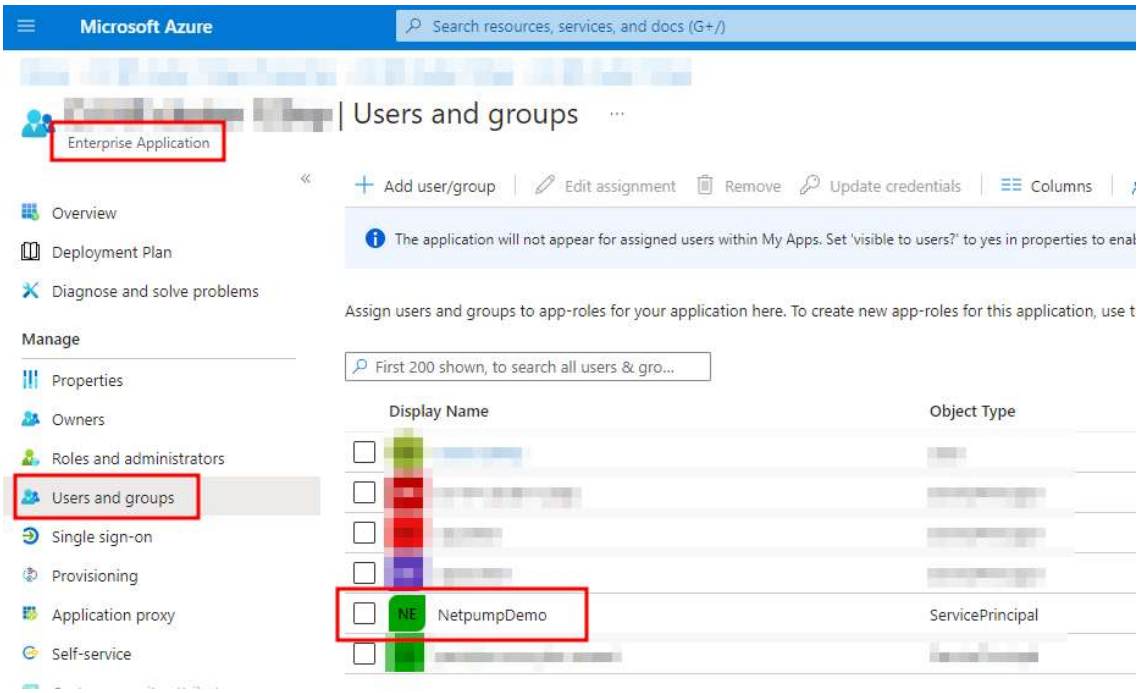


3. Go to the newly created Automation Account

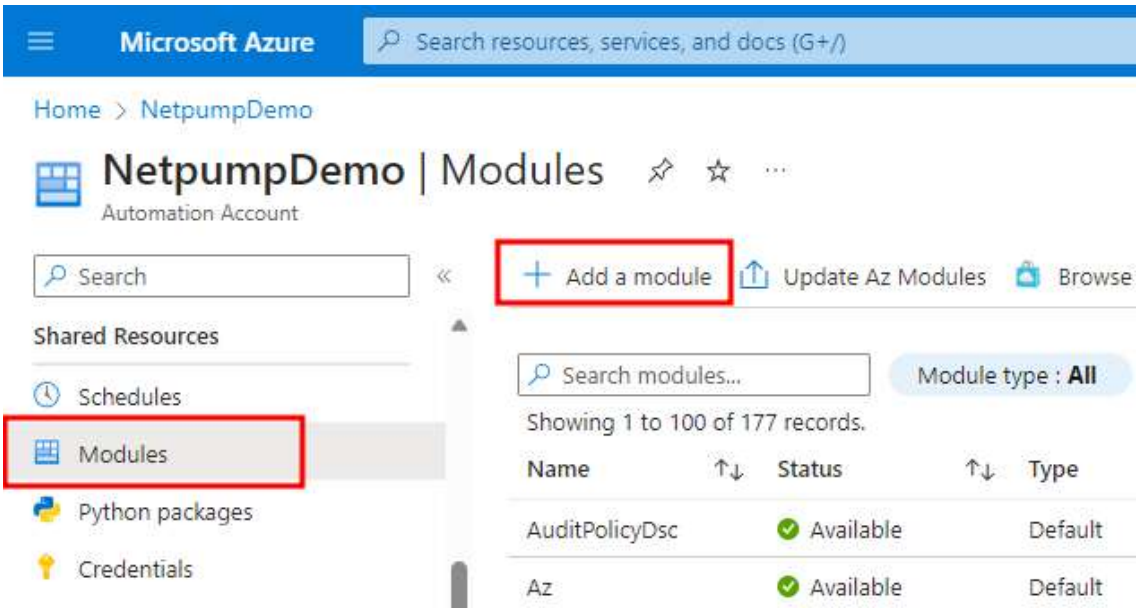
4. Click in the Identity section in the left menu and copy the Object (principal) ID to the clipboard



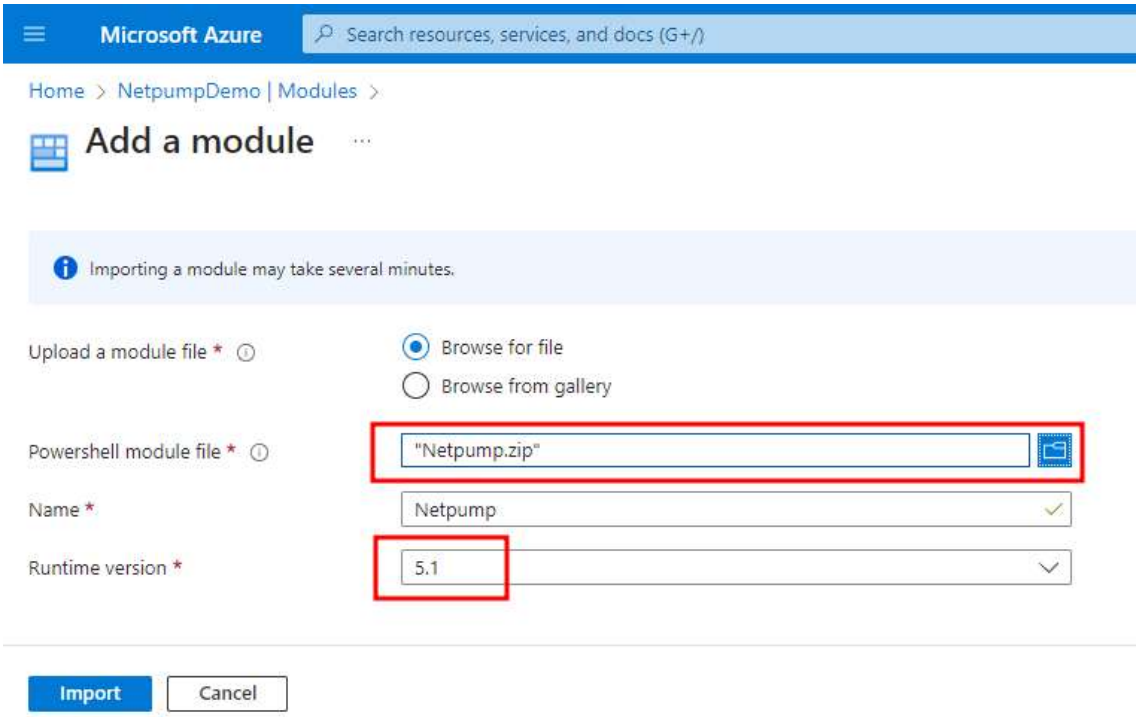
5. Run PowerShell on your local computer
6. At the PowerShell prompt, run Connect-MgGraph -TenantId $tenantId -Scopes "User.Read","Application.ReadWrite.All"
7. At the PowerShell prompt, run New-MgServicePrincipalAppRoleAssignment -ServicePrincipalId $SERVICE_PRINCIPAL_OBJECT_ID -PrincipalId $SERVICE_PRINCIPAL_OBJECT_ID -ResourceId $NETPUMP_ENTERPRISE_APP_OBJECT_ID -AppRoleId $AUTOMATION_APP_ROLE_ID. Replace each of the $VARIABLEs in this command with your actual values, which will all be GUIDs in the format 00000000-0000-0000-0000-000000000000.
8. To confirm that the above command was successful, check the Enterprise Application for your Netpump cluster, on the Users and groups page, the name of your Automation Account should now appear in the list.



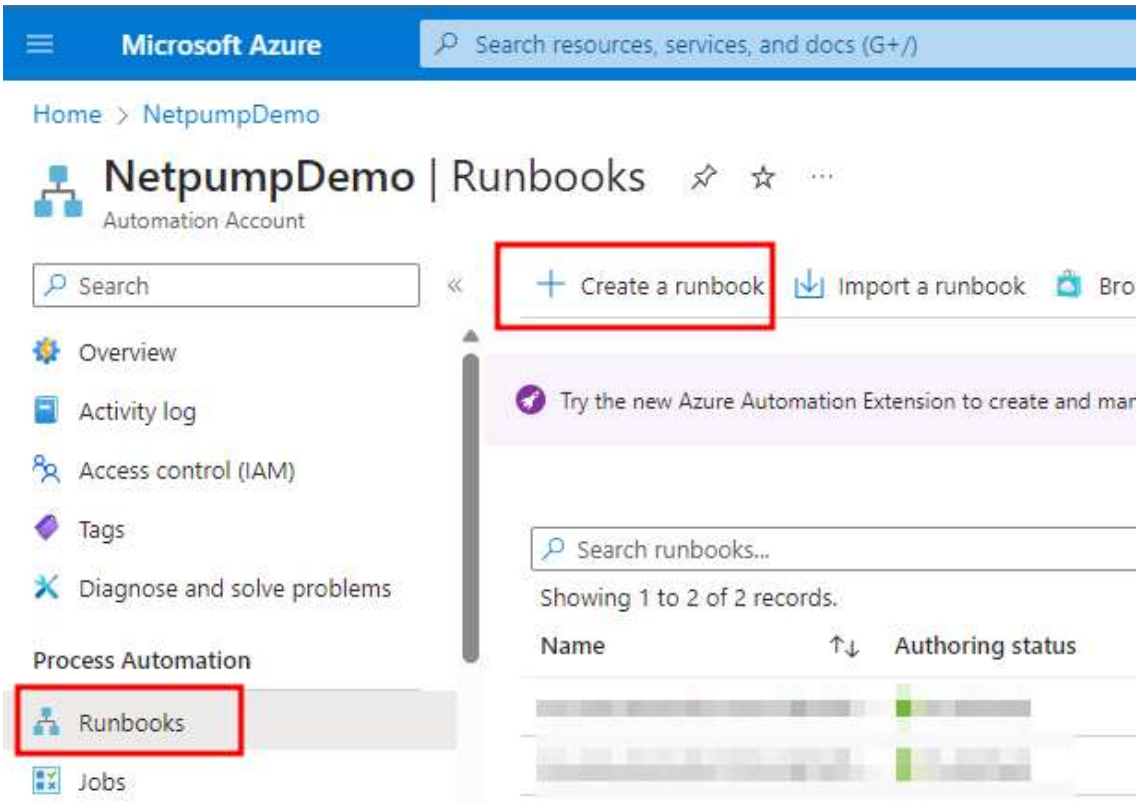
9. Back in the Automation Account, click in the Modules section in the left menu and click Add a Module



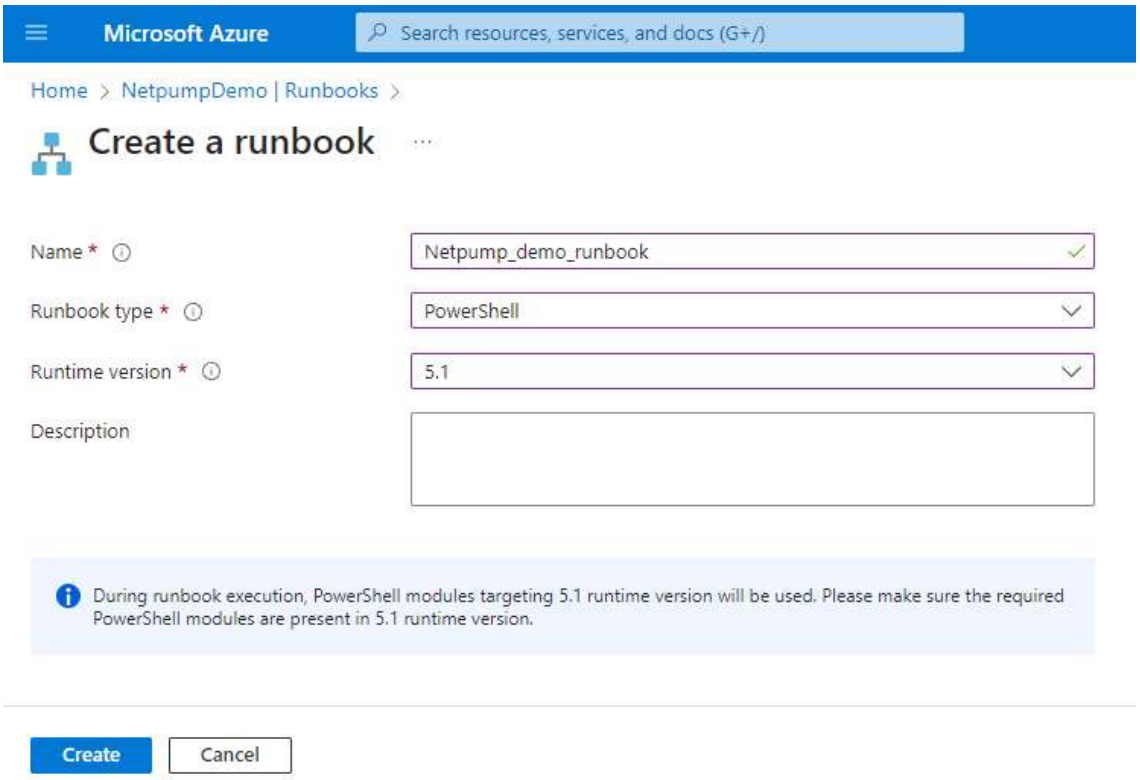
10. Upload the Netpump.zip file and choose 5.1 for the PowerShell version. Click Import.



11. Go to the Runbooks section in the left menu and click Create a runbook.



12. Choose a name for the runbook, type of PowerShell, version 5.1, and click Create.



13. Edit the Runbook and enter the following script, replacing the parameters with your origin and destination server URLs and your origin and destination UNC paths:

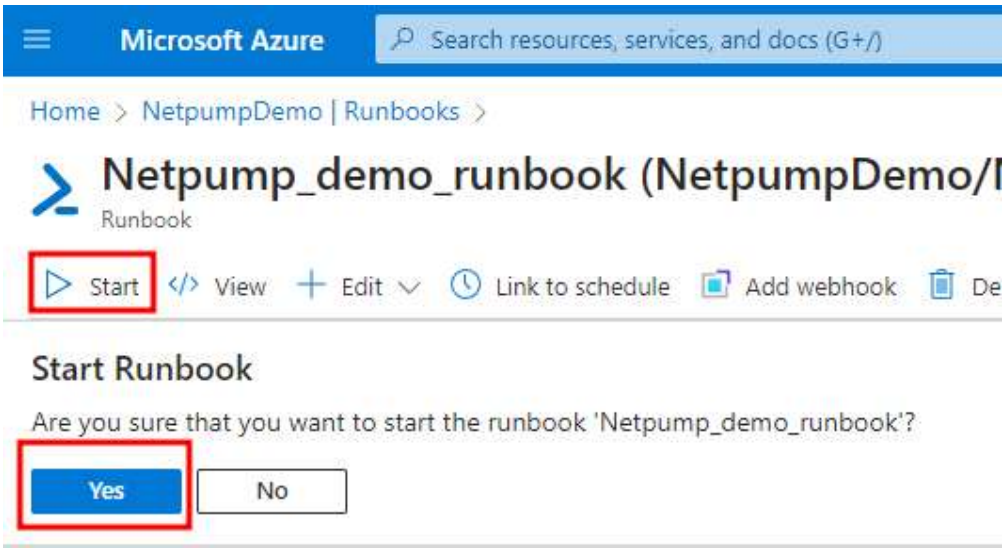
Connect-AzAccount -Identity

Import-Module Netpump

Start-NetpumpTransfer -DestinationService https://YOUR-URL -DestinationFolder \\YOUR-UNC\YOUR-SHARE\YOUR-FOLDER -OriginService https://YOUR-URL -Paths \\YOUR-UNC\YOUR-SHARE\YOUR-SOURCE-FILE

14. Click Publish

15. Click Start, and Yes to confirm



16. Your file is copied from the source to the destination via Netpump

16. Post-Deployment Validation – Confirm Everything is Working

Now that the deployment and configuration are done, perform a few checks to ensure the Netpump Data deployment is healthy:

Check	**Expected Result**
Netpump server status endpoint	Visiting `https://pump1.<your-domain>/status` (and `pump2`) returns a JSON response, e.g. `{ "ready": true }`, indicating the server is up and ready. *(You may do this in a browser; you might get a basic auth prompt or need to ignore the cert warning again.)*
Test file transfer (Desktop or API)	A file transfer via the Netpump system completes successfully and noticeably faster (e.g., ~20× faster) than a traditional direct SMB transfer of the same file. This demonstrates the acceleration is in effect.
Azure Portal –Resource health (for the Netpump VMs)	In the Azure Portal, navigate to the Resource health for the Netpump VMs or the managed application. All Netpump virtual machines should show as **Available/Healthy** (no failures or critical alerts).

If all the above checks are good, your Netpump Data deployment is successfully up and running!

If any check fails or if you encounter errors at any step:

- **Revisit your configurations**: The most common culprits are an incorrect **Client Secret**, an incorrect **Certificate URL**, or a wrong **Storage account key**. Even a single character off will cause authentication to fail.
- Make sure the Azure AD apps have the correct permissions and consent. If the status endpoint isn't ready, it might be an auth issue – check the application's log (if accessible) or re-run the config step and consent.
- You can also redeploy or rotate secrets if needed (for example, generate a new client secret and update the config) in case a secret was lost.

17. Where to Get Help

If you need further assistance or run into issues that you cannot resolve, use these resources:

- **Pacbyte Support Slack**: Join the #netpump-azure channel on Slack (if provided by your organization or the vendor) to ask questions to the support team and community.
- **Netpump Data Documentation**: Refer to the official docs at the Netpump Data GitHub Docs for more detailed guides, troubleshooting tips, and updates.
- **EULA and Additional Info**: Review the End User License Agreement (EULA) and additional notes in the documentation (e.g., the file Netpump-EULA.pdf in the docs repository) to understand usage terms.

Congratulations! You have deployed Netpump Data on Azure and configured it end-to-end. Your new data transfer service is ready to use. Enjoy significantly faster data transfers and remember to monitor the system and rotate secrets/certificates as needed for security maintenance. If you ever need to redo the setup or add another Netpump cluster, you can follow this guide again with new app registrations and resources (consider using a new prefix for multiple deployments). Happy pumping!