

Infodash Development Environment Guidance

Many firms would like a development environment where they can test new Infodash versions before upgrading.

There are two primary methods to creating a development environment:

[Shared Development Environment](#)

[Isolated Development Environment](#)

[What Are Infodash Global Pointers?](#)

[Infodash Global 'Pointers'](#)

Shared Development Environment [🔗](#)

A shared development environment is an environment where Infodash has two instances (DEV/PROD) that run under the same App Service Plan.

✓ PROS

- You only have to pay for a single app service plan.
- Both applications run inside the same app service but allow for separate configurations.

⚠️ CONS

- If you do something against the DEV instance, it can negatively affect the PROD instance since the resources (RAM/MEMORY/DTU) are shared.
 - You can scale your app service and app service plan to ensure shared resources do not become a bottleneck. In most cases, this is not an issue and we have never had an actual case of a development slot affecting the production slot.

A shared development environment is accomplished by using App Service Slots. Slots are different deployment environments (e.g. **staging, development, production**) for your application within the **same** app service. These slots are essentially "live" versions of your app that run independently of your main (or "production") app.

📌 Note

Each **slot** can have its own Infodash database and its own configuration.

Here are some key details about App Service Slots.

App Service 'Slots' [🔗](#)

App Service Slots are a feature in **Azure App Service** that allow you to host multiple versions of your web application in separate deployment environments (called *slots*) within the same App Service Plan. Each slot behaves like a separate app instance with its own configuration, URL, and environment, but they share the same underlying infrastructure.

Key Features of App Service Slots: [🔗](#)

1. **Staging and Testing:** You can use deployment slots to test your app in a staging environment before moving it to production.
2. **Swapping Slots:** The primary feature of App Service Slots is the ability to perform a "slot swap." This allows you to promote a version of your app from a staging slot to production with no downtime, by swapping the slots. During the swap:

- The staging slot becomes the production slot.
 - Traffic is routed to the new production slot.
 - Configuration settings (like app settings and connection strings) can optionally swap as well.
3. **Zero-Downtime Deployments:** By deploying to a non-production slot (e.g., staging), you can avoid downtime during updates. After testing, you can swap the slots to move the update into production seamlessly.
 4. **Separate URLs for Each Slot:** Each slot has a unique URL, allowing you to test different versions of your app before making them live. The default production slot will have the default URL of the app service (e.g., `myapp.azurewebsites.net`), while additional slots will have a suffix like `myapp-staging.azurewebsites.net`.
 5. **Slot-Specific Configurations:**
 - Slots allow you to configure settings independently, such as environment variables, connection strings, or feature flags, which may differ between production and staging environments.
 - Certain settings (e.g., scale settings) are shared between all slots.
 6. **Rollback to Previous Version:** If you deploy a new version of your app to production and encounter issues, you can quickly roll back by swapping back to the previous slot.

How to setup a development environment (Shared) [🔗](#)

As of version 4.1.6.0 of Infodash, new installations get a development slot created for them automatically. If you do not have a development slot, follow the instructions below:

1. Open your App Service in [Azure](#). e.g. *infodashapi-company*
2. Click on **Deployment > Deployment Slots** via the left-hand navigation.
3. Click on **Add** to create a new slot.
 - a.

Add Slot ✕

Name

 infodash-collaboration-development.azurewebsites.net

Clone settings from:

4. For the **name** field, use **dev** or **development**. This will shared the same base url of your app service, such as **infodashapi-company** and it will append the slot name to create a unique URLs. So, if you use **dev** as the slot name, the dev app service slot would be **infodashapi-company-dev.azurewebsites.net**.
5. Make sure you clone your settings from the **Production** slot. This will ensure all of your environment variables and connection strings are cloned.

Create a new Infodash database [🔗](#)

The ensure your Infodash development slot can have its own configuration and sites, you need to create another Infodash database. If your database is running on-premise, you can simply back-up and restore the current database with a new name such as **infodash-dev**.

If your Infodash database is running in Azure (default since 4.0.0.0), you can create a new database via the [SQL Server](#).

Note

Doing this in Azure will create a blank database

Update your connection strings

After you have created your new slot and dedicated database, you need to update the connection strings to point to the new Infodash database (and any other connection strings).

1. Open your App Service in [Azure](#). e.g. *infodashapi-company*
2. Click on **Settings > Environment Variables > Connection Strings**.
3. Edit each connection string to point to the respective **development** databases.
4. Ensure the service account credentials are still valid for these databases.

Note

For the Infodash database connection, the service account must have **dbo** rights. For all other databases, we only require **datareader**. If **datareader** is defined, you must also grant the service account for this connection **EXECUTE** rights on any **Infodash_xxx_xxx** stored procedure.

```
1 USE <DatabaseName>;
2 GRANT EXECUTE ON OBJECT::<StoredProcedureName>
3     TO <SqlAccount>;
```


Isolated Development Environment

PROS

- Two app services and two app service plans allow for 100% isolation from each other.
- One app cannot impact the other.

CONS

- Additional cost to spin up another app service plan.

 An isolated environment requires two separate installations. One for each app service. This is not performed as part of your initial install and will require professional services to deploy a new app service and setup.

What Are Infodash Global Pointers?

Global Pointers are how Infodash decides which environment your SharePoint site should connect to — either the **Production (PROD)** environment or a **Development (DEV/TEST)** version.

Think of them as **address cards** for your SharePoint site. By default, all sites use the **main address** (Production). But if you want a specific site to connect to a development version of Infodash, you can give it a different address (pointer) in the configuration.

Why Does This Matter?

Using pointers lets your organization:

- **Test changes safely** without affecting live users.

- Keep DEV and PROD data, configuration, and updates completely separate.
- Upgrade or try new Infodash features on a few test sites before rolling them out company-wide.

How to Check If Your Site is Using DEV or PROD [↗](#)

Here's a simple way a **non-technical user** or admin can check:

1. Look at the Site URL in your browser:

- If you are on something like `https://infodashdemo.sharepoint.com/sites/development` or a similarly named "test" site, there's a good chance it's pointing to the **DEV version**.
- If it's your normal business SharePoint site (e.g., `/sites/hr`, `/sites/finance`), it's likely using **PROD**.

2. Use the Command (IT/Admins Only):

If you have access to PowerShell and PnP modules, you can run:

```
1 Get-PnPStorageEntity -Key infodashcore
2
```

This command will return a list of configured pointers. If your SharePoint site is listed explicitly, it's using a specific DEV/TEST version. If it's not listed, it defaults to PROD.

How the Configuration Works [↗](#)

Here's a **real-world example**:

```
1 {
2   "default": {
3     "uri": "https://infodash-collaboration.azurewebsites.net",
4     "appId": "852d84a5-1461-4a1e-ac20-c8512b148e36"
5   },
6   "https://infodashdemo.sharepoint.com/sites/development": {
7     "uri": "https://infodash-collaboration-dev.azurewebsites.net",
8     "appId": "f81636e9-f600-43b0-9833-ddcace1a2d5a"
9   }
10 }
11
```

This means:

- All sites **except** the `/development` one will use PROD.
- The `/development` site uses the DEV version of Infodash.

Important Consideration for Admins [↗](#)

To avoid accidentally updating **global tenant-level Infodash apps from DEV**, always make sure this setting is in place on non-production environments:

- **Set this App Setting in Azure** for the DEV slot:
 - `PortalConfig:IsGlobalTenant = false`

This ensures DEV can only update **individual site catalogs**, not the entire organization's Infodash instance.

Summary for Non-Technical Users [↗](#)

- If you're working in a **test or dev** SharePoint site, you might be connected to a **separate Infodash instance**.
- These environments allow safe testing without breaking anything in production.

- Behind the scenes, this is controlled with "global pointers" — like routing signs telling your SharePoint where to go.
- If you're not sure, contact your IT team or check with the SharePoint admins.

Infodash Global 'Pointers' [🔗](#)

Infodash uses SharePoint global storage entities to tell SharePoint sites where to find the appropriate endpoint. Using pointers, you can tell Infodash to use the **default** (PROD) API for all sites and then use additional pointers to override the default and tell DEV/TEST sites. Let's take a look.

By Default, your global storage entity of **infodashcore** has a single entry of **default**. Meaning, every Infodash Site would look to this API for its configuration, integrations and settings.

```
1 {
2   "default": {
3     "uri": "https://infodash-collaboration.azurewebsites.net",
4     "appId": "852d84a5-1461-4a1e-ac20-c8512b148e36"
5   }
6 }
```

To separate the sites to match your prod/development separation, we need to update the global storage entity. Each site that needs to point to a different API must be defined in the storage entity. Example below.

```
1 {
2   "default": {
3     "uri": "https://infodash-collaboration.azurewebsites.net",
4     "appId": "852d84a5-1461-4a1e-ac20-c8512b148e36"
5   },
6   "https://infodashdemo.sharepoint.com/sites/development": {
7     "uri": "https://infodash-collaboration-dev.azurewebsites.net",
8     "appId": "f81636e9-f600-43b0-9833-ddcace1a2d5a"
9   }
10 }
```

Looking at the above configuration, our global storage entity is configured as follows:

```
1 Get-PnPStorageEntity -Key infodashcore
```

- The site `https://infodashdemo.sharepoint.com/sites/development` will use `https://infodash-collaboration-dev.azurewebsites.net` as its API and Admin Center.
- **All other** sites will use the **default** (prod) URL as its API and Admin Center.

Using this approach, different SharePoint sites can use different Infodash Applications, allowing you to create 100% separation between environments.

⚠ Important Notes

When you have multiple slots deployed on your app service and your tenant is configured for **Global Deploy** (environment variables of API), each instance of the API has the capability of updating the global tenant version of the Infodash Application. To prevent this, you should always set your DEV/TEST API's **PortalConfig:IsGlobalTenant** environment variable to false. That way, the non-prod instances can only update individual site app catalogs and not the tenant app catalog.

You can update this setting via the appropriate slot in the App Service under **Settings > Environment Variables > PortalConfig:IsGlobalTenant** by setting it to **False**.

This is only required if you want to run a different version of Infodash in your development sites than your production sites.