TOC

Logi Symphony 24

Install, Upgrade, and Remove Symphony 24

Install Symphony - Kubernetes

Adding Logi Symphony to your environment involves installing, configuring, and licensing the components you'll use as a stand alone or embedded resource. You may need to install multiple components that work together to provide your desired analytics experience, depending on your licensing arrangements.

Important: To install Symphony in a Kubernetes environment, you'll need an intermediate level of familiarity and experience with Kubernetes. See https://kubernetes.io/docs/home/.

Important: Download the latest installation package and the complete readme at https://repo-logisymphony.insightsoftware.com/helm-charts/.

Prerequisites

- A Kubernetes cluster of appropriate compute availability
- The Helm CLI installed
- Your command line tool, kubectl, is installed and configured to use with your cluster
- Know your selection of modules you wish to install:
  - Full Symphony 24
  - Symphony 24 with Logi AI (beta)
  - Symphony with Visual Data Discovery only.
  - Symphony with Managed Dashboards and Reports only.

Kubernetes Cluster Requirements

There are no specific limits in the Helm chart by default, but we suggest the following compute availability on your cluster:

Recommended Compute Availability

- 10 Cores
- 24 Gi Memory

Required for Scheduling by Kubernetes

- 4 Cores
- 16 Gi Memory

> **Important:** These specifications are intended as an initial installation guide. Your environment's specific workloads may demand a higher number of cores and more memory available for computing.

Other Requirements

The memory requirements above do not take into account other components that are a part of Logi Symphony. These components are external to Helm chart requirements. Size your environment following the recommendations for each component. These components include:

- PostgreSQL
- RabbitMQ

Installation Steps - All Modules

- Step 1: Install a Proxy or Ingress Controller
- Step 2: Determine Your URL
- Step 3: Add The Symphony Helm Repository
- Step 4: Create A Symphony Values.yaml File
- Step 5: Install Logi Symphony
- Step 6: Get The Pods For Your Namespace

For information about accessing Symphony after it is installed, see Access ComposerSymphony and Log Into the ComposerSymphony UI.

For information about accessing Symphony after it is installed, see Access Symphony and Log Into the ComposerSymphony UI (Getting Started) or Log Into the Symphony UI (full Symphony documentation).

Step 1: Install a Proxy or Ingress Controller

Install the proxy or ingress controller you want to use to provide flexible control over incoming traffic to your Symphony environment.

One option is to install a NGINX Ingress Controller as in the example provided here.

Note: For other install methods for the ingress controller, see https://bitnami.com/stack/nginx-ingress-controller/helm.

Install a NGINX Ingress Controller

1. Install NGINX using the following commands:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo update
helm install my-ingress-controller bitnami/nginx-ingress-controller
```

2. Verify your ingress controller by running the following command to return all services in the namespace:

```
kubectl get services
```

3. If the EXTERNAL-IP is set to a value and not stuck in a pending space, the ingress controller is exposed correctly. You can additionally verify this by visiting the external IP address: you should be served an NGINX 404 page.

Step 2: Determine Your URL

You have two options to choose from for your environment's URL:

- Use the EXTERNAL-IP returned in the previous step
- Use the URL from your cluster

Step 3: Add the Symphony Helm Repository

Install the Helm chart with the release name of logisymphony1 on your Kubernetes cluster using default configuration settings. Adjust parameters as needed.

```
helm repo add insightsoftware https://repo-logisymphony.insightsoftware.com/helm-charts/
helm repo update
helm install logisymphony1 insightsoftware/logisymphony
```

Uninstall the Helm Chart

If you need to uninstall the Helm chart, run this command to remove all Kubernetes components associated with the chart and delete Symphony.

```
helm delete logisymphony1
```

Step 4: Create a Symphony `values.yaml` File

Use Helm to generate a `values.yaml` file:

```
helm show values insightsoftware/logisymphony
```

or

```
helm show values insightsoftware/logisymphony > values.yaml
```

Next, edit in an updated Admin user password, and the ingress information.

Set the Admin Password and Enable Features

Set a strong password: a combination of at least nine characters that includes upper and lower case letters, numbers, and special characters. Enable and disable features as needed:

```
# Default values for Logi Symphony.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

global:
  logi:
    symphony:

      # Use an existing secret to provide all Logi Symphony related secrets.
      # It will use a single secret, and will expect specific keys to available in
```

```
      # the secret.  See the helm readme for more details on using the value.
      existingSecret: ""

      # Set the Composer, and Dundas BI admin credentials.
      auth:
        adminPassword: "ChangeMe!123"
      # If the above section is not set, then the admin password is
      # set through the .values.managed, or .values.discovery sections.

      managed:
        # The managed component cannot be completely be disabled, however setting enabled to false will result in running
        # the managed component in a minimal mode.
        enabled: true

      discovery:
        # Disable the data discovery component.
        enabled: true

      rabbitmq:
        # Disables the rabbitmq component.
        enabled: true

      ai:
        # Enables the BETA AI feature.
        enabled: false
```

Enable the Data Gateways

To enable data gateways, add to the respective modules as needed.

Visual Data Discovery

```
discovery:
  # Data Gateway Service configuration
  dataGatewayService:
    # Is Data Gateway Service enabled
    enabled: true
```

Managed Dashboards and Reports

```
mananged:
  dundas:
    bi:
      gatewayhub:
        # enables the gatewayhub deployment.
        enabled: true

        # The gatewayhub service.
        service:
          enabled: true
```

Managed Service Values

Set the Managed service values using the `managed` section of `values.yaml`.

```
managed:
# Default values for dundas-bi.
  dundas:
    bi:
```

Data Discovery Service Values

Set the Data Discovery service values using the `discovery` section of `values.yaml`.

```
discovery:
  zoomdataClient:
```

Symphony Databases

If `global.postgresql.enabled` is `true`, set the Postgres authentication details in the `global.postgresql.auth` section.

If `global.postgresql.enabled` is `false`, set the database information in the `managed` and `discovery` sections of the `values.yaml`. See the Managed Dashboards and Managed Reports helm readme and the Visual Data Discovery helm readme embedded in their respective helm charts.

The other postgres details are set in the `postgresql` section of the `values.yaml` as seen in the following example. Add databases for installed modules as needed.

```
postgresql:
  # Is PostgreSQL database enabled
```

```
  enabled: true
  image:
    # Image tag
    tag: 12
  primary:
    initdb:
      # Username to execute the initdb scripts
      user: "postgres"
      # Password to execute the initdb scripts
      password: "LogiSymphony!123456"
      # Scripts to be run at first boot
        scripts:
          "initdbScript.sql": |
            CREATE DATABASE "zoomdata" WITH OWNER logisymphonyuser;
            CREATE DATABASE "zoomdata-upload" WITH OWNER logisymphonyuser;
            CREATE DATABASE "zoomdata-keyset" WITH OWNER logisymphonyuser;
            CREATE DATABASE "zoomdata-user-auditing" WITH OWNER logisymphonyuser;
            CREATE DATABASE "zoomdata-qe" WITH OWNER logisymphonyuser;
```

Note: To add more databases for the Visual Data Discovery service , create them in this section along with the postgres database.

Data Discovery

Include this `initdbScript.sql` with appropriate `OWNER` information for the Visual Data Discovery module.

```
CREATE DATABASE "zoomdata" WITH OWNER logisymphonyuser;
CREATE DATABASE "zoomdata-upload" WITH OWNER logisymphonyuser;
CREATE DATABASE "zoomdata-keyset" WITH OWNER logisymphonyuser;
CREATE DATABASE "zoomdata-user-auditing" WITH OWNER logisymphonyuser;
CREATE DATABASE "zoomdata-qe" WITH OWNER logisymphonyuser;
```

Managed Dashboards and Reports

Include this `initdbScript.sql` with appropriate `OWNER` information for the Managed Dashboards and Reports modules.

```
CREATE DATABASE "dundasbi" WITH OWNER test;
\connect "dundasbi";
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
CREATE EXTENSION IF NOT EXISTS "unaccent";
CREATE DATABASE "dundasbiwh" WITH OWNER test;
```

```
\connect "dundasbiwh";
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
CREATE EXTENSION IF NOT EXISTS "unaccent";
```

Logi AI

Include this `initdbScript.sql` with appropriate `OWNER` information for the Logi AI module.

> **Important:** This feature is considered to be released in beta for your testing purposes. Workflows and features may change before a production-ready version is released.

```
CREATE DATABASE "LogiSymphonyAI" WITH OWNER test;
\connect "LogiSymphonyAI";
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
```

Data Discovery CORS Whitelisting

If you're embedding your Visual Data Discovery content, you must add the appropriate domain names to whitelisting for CORS.

```
   properties:
     access.control.allow.origin: "example1.com,example2.com"
```

RabbitMQ Deployment

By default this helm chart will deploy a RabbitMQ helm chart. The RabbitMQ configuration is passed down to sub charts Visual Data Discovery, and Managed Dashboards and Managed Reports via Environment variables set directly in the `values.yaml`:

```
discovery:
  zoomdataWeb:
    extraEnvs:
      # RabbitMQ configuration belongs here.
managed:
  dundas:
    bi:
      setup:
        extraEnvs:
        # RabbitMQ configuration belongs here.
```

Important: RabbitMQ is required. If you prefer to use it outside of the Symphony helm chart, set `global.logi.symphony.rabbitmq.enabled` to `false` and define the environment variables above to point to your accessible RabbitMQ instance.

Important: Currently only the values set by default in `values.yaml` are supported.

Configure RabbitMQ

You can initialize RabbitMQ with Exchanges, Bindings, and Queues in two different ways.

Method one (default): Pass the following Environment variable. With this set, RabbitMQ populates with the required exchanges, bindings, and queues during the handledatabase container as part of the Managed Dashboards and Managed Reports start up.

```
managed:
  dundas:
    bi:
      setup:
        - name: INIT_RABBITMQ_FOR_SYMPHONY
          value: "true"
```

Method two: Pass the configuration information as part of configuration.

```
rabbitmq:
  extraSecrets:
    load-definition:
      load_definition.json: |
        {
          "vhosts": [
          {
            "name": "symphony"
          }
          ],
          "users": [
          {
            "name": "{{ .Values.auth.username }}",
            "password": "{{ .Values.auth.password }}",
            "tags": ["administrator"]
          }
          ],
          "permissions": [
```

```
{
    "user": "{{ .Values.auth.username }}",
    "vhost": "symphony",
    "configure": ".*",
    "write": ".*",
    "read": ".*"
  }
],
"exchanges": [
  {
    "name": "composer.source",
    "vhost": "symphony",
    "type": "topic",
    "durable": true,
    "auto_delete": false,
    "internal": false,
    "arguments": {}
  },
  {
    "name": "dundas.broadcast",
    "vhost": "symphony",
    "type": "fanout",
    "durable": true,
    "auto_delete": false,
    "internal": false,
    "arguments": {}
  }
],
"queues": [
  {
    "name": "create-native-structure",
    "vhost": "symphony",
    "durable": true,
    "auto_delete": false,
    "arguments": {}
  },
  {
    "name": "update-native-structure",
    "vhost": "symphony",
    "durable": true,
    "auto_delete": false,
    "arguments": {}
  },
  {
```

```
      "name": "delete-native-structure",
      "vhost": "symphony",
      "durable": true,
      "auto_delete": false,
      "arguments": {}
    },
    {
      "name": "create-native-structure-column",
      "vhost": "symphony",
      "durable": true,
      "auto_delete": false,
      "arguments": {}
    },
    {
      "name": "update-native-structure-column",
      "vhost": "symphony",
      "durable": true,
      "auto_delete": false,
      "arguments": {}
    },
    {
      "name": "delete-native-structure-column",
      "vhost": "symphony",
      "durable": true,
      "auto_delete": false,
      "arguments": {}
    }
  ],
  "bindings": [
    {
      "source": "composer.source",
      "vhost": "symphony",
      "destination": "create-native-structure",
      "destination_type": "queue",
      "routing_key": "source.created",
      "arguments": {}
    },
    {
      "source": "composer.source",
      "vhost": "symphony",
      "destination": "update-native-structure",
      "destination_type": "queue",
      "routing_key": "source.updated",
      "arguments": {}
```

```
            },
            {
              "source": "composer.source",
              "vhost": "symphony",
              "destination": "delete-native-structure",
              "destination_type": "queue",
              "routing_key": "source.deleted",
              "arguments": {}
            },
            {
              "source": "composer.source",
              "vhost": "symphony",
              "destination": "create-native-structure-column",
              "destination_type": "queue",
              "routing_key": "source.field.created",
              "arguments": {}
            },
            {
              "source": "composer.source",
              "vhost": "symphony",
              "destination": "update-native-structure-column",
              "destination_type": "queue",
              "routing_key": "source.field.updated",
              "arguments": {}
            },
            {
              "source": "composer.source",
              "vhost": "symphony",
              "destination": "delete-native-structure-column",
              "destination_type": "queue",
              "routing_key": "source.field.deleted",
              "arguments": {}
            }
          ]
        }
loadDefinition:
  enabled: true
  existingSecret: load-definition
extraConfiguration: |
  default_vhost = symphony
  load_definitions = /app/load_definition.json
```

Ingress Updates

Modify the ingress section by defining the URL from Step 2, and uncomment any lines until your file matches the commands indicated below.

```
ingress:
  enabled: true

  appendToPath: ""

  annotations:
    kubernetes.io/ingress.class: nginx
    kubernetes.io/tls-acme: "true"
    nginx.ingress.kubernetes.io/proxy-http-version: "1.1"
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
    nginx.ingress.kubernetes.io/affinity: "cookie"
    nginx.ingress.kubernetes.io/session-cookie-name: "dbiroute"
    nginx.ingress.kubernetes.io/session-cookie-expires: "172800"
    nginx.ingress.kubernetes.io/session-cookie-max-age: "172800"
    nginx.ingress.kubernetes.io/session-cookie-path: /
    nginx.ingress.kubernetes.io/proxy-connect-timeout: "86400"
    nginx.ingress.kubernetes.io/proxy-read-timeout: "86400"
    nginx.ingress.kubernetes.io/proxy-send-timeout: "86400"
    nginx.ingress.kubernetes.io/proxy-body-size: "500m"
  hosts:
    - host: <fully-qualified-domain-name>
    paths:
      - /
```

You must enable sticky sessions for a load balanced implementation of the managed service in Logi Symphony. This example includes annotations to add to enable sticky sessions with a NGINX ingress controller.

```
ingress:
  enabled: true
  annotations:
    kubernetes.io/ingress.class: nginx
    kubernetes.io/tls-acme: "true"
    nginx.ingress.kubernetes.io/proxy-http-version: "1.1"
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
    nginx.ingress.kubernetes.io/affinity: "cookie"
    nginx.ingress.kubernetes.io/session-cookie-name: "symphonyroute"
    nginx.ingress.kubernetes.io/session-cookie-expires: "172800"
    nginx.ingress.kubernetes.io/session-cookie-max-age: "172800"
    nginx.ingress.kubernetes.io/session-cookie-path: /
    nginx.ingress.kubernetes.io/proxy-connect-timeout: "86400"
    nginx.ingress.kubernetes.io/proxy-read-timeout: "86400"
    nginx.ingress.kubernetes.io/proxy-send-timeout: "86400"
```

```
    nginx.ingress.kubernetes.io/proxy-body-size: "500m"
    nginx.ingress.kubernetes.io/server-snippet: |
      gzip on;
      gzip_types = text/plain image/svg+xml application/xml text/css application/javascript application/json
"application/json;charset=utf-8" application/font-woff;
```

Alternatively, deploy a redis subchart:

```
managed:
  dundas:
    bi:
      # Dundas BI Cache Provider
      cache:
        # It is also possible to use a cache provider outside of the helm chart, and is the recommended way for Production.
        redis:
          # Enables a Redis subchart.  See the Redis subsection for more helm values.
          enabled: true

            # If password is empty a random one will be chosen.
            password: ""
```

SSL Set Up

Use an ingress to enable SSL with the Symphony Helm chart. First create a secret with your certificate information.

```
kubectl create secret tls exampledomain --key C:\temp\exampledomain.keyfile --cert C:\temp\exampledomain.certfile
```

Next, apply that secret to the ingress inside the tls.

```
ingress:
  ...
  hosts:
    - host: exampledomain.com
      paths:
        - /
```

End readme version of instructions

Step 5: Install Logi Symphony

With the previous steps completed, run the following command:

```
helm install logisymphony1 insightsoftware/logisymphony -f values.yaml
```

This installs Symphony. When it starts, proceed to the next step.

Step 6: Get the Pods for Your Namespace

Run this command to return a list of all pods running.

```
kubectl get pods -A
```

Install Symphony - Windows

Adding Logi Symphony to your environment involves installing, configuring, and licensing the components you'll use as a stand alone or embedded resource. You may need to install multiple components that work together to provide your desired analytics experience, depending on your licensing arrangements.

Windows support for Symphony includes one Symphony instance per Windows instance, and includes the Getting Started tutorial.

Important: Download the latest installation package here: https://s3.us-west-1.amazonaws.com/installer.logisymphony.com/windows/Latest/Logi.Symphony.Setup.exe.

Prerequisites

- Windows 10, 11, and Windows Server 2016, 2019, or 2022 (64-bit) or higher.

  Note: Disable Internet Explorer Enhanced Security in Windows Server 2016 to avoid multiple Internet Explorer security notifications.

- Know the selection of modules you wish to install:
  - Symphony with Visual Data Discovery, Managed Dashboards, Managed Reports.
  - Symphony with Managed Dashboards and Managed Reports only.

    Note: Logi AI is not supported on Windows.

- An email address to assign to the Admin user and an appropriate password.
- Appropriate names, passwords, descriptions, and port information for creating new databases or connecting to existing databases.

- If you are performing a silent installation, you must install PostgreSQL first if required by your environment before you install Symphony.
- Download the installer: https://s3.us-west-1.amazonaws.com/installer.logisymphony.com/windows/Latest/Logi.Symphony.Setup.exe.

Windows Requirements

- 4 Cores
- 16 Gi Memory minimum. Add more to support multiple data connectors.
- 40+ Gi of storage

Other Requirements

- Visual Data Discovery Module: PostgreSQL. Install before you begin Symphony installation (required for silent install), or as part of the installation process.
- Managed Dashboards and Reports Modules: You can use PostgreSQL, or an existing Microsoft SQL server.
- .NET 8.0 Desktop Runtime or later - Install before you begin Symphony installation. You will be prompted to install during Symphony deployment if it is not present.
- 16 Gi Memory minimum. Add more to support multiple data connectors.
- 40+ Gi of storage.

Important: These specifications are intended as an initial installation guide. Your environment's specific workloads may demand a higher number of cores and more memory available for computing.

Configuration Specific Requirements

- RabbitMQ - Allows for the integration of data connectors among installed modules. Install before you begin, or as part of the installation process.
- Symphony application and management databases - Connect existing MS SQL or PostgreSQL databases, or prompt Symphony to install PostgreSQL as

needed.

- Python - Provides Python transforms and a Python Machine learning provider in the Managed Dashboards and Reports module.

Note: The Deployment Console lets you launch the Deployment Console Wizard and check your version, get the silent install, install export configs, or call with a deployment config file to do a silent install.

Installation Steps - All Modules

Step 1: Download and Launch Installation Packages

1. Download and install .NET 8.0 Desktop Runtime or later if it is not already present.
2. Download the Symphony setup package.
3. Launch the application as a Windows administrator. A Logi Symphony Setup work area opens.
4. Enter an installation path, or accept the default path provided.
5. Select **Install Setup Files** to install the files you need to install the deployment console wizard.

   Important: If Symphony installs a PostgreSQL database for you as part of this process, the user name is `postgres`.

6. Once complete, select **Launch Application** to launch the **Logi Symphony Deployment Wizard**.

   Note: You can access the deployment wizard later from the Windows Start Menu.

Step 2: Select Modules and Install Prerequisites

1. Select the option to **Add Instance** from the wizard by typing in the appropriate option number, then select Enter.
2. By default, the Managed and Discovery modules are selected for installation. Enter a number for a module to toggle installation on and off for your modules as desired, then type `y` to continue installation.

> **Important:** If you only install the Managed Dashboards and Reports module as a way to migrate your Dundas BI instance, you can alternatively use your existing SQL server instead of installing PostgreSQL. If you install Visual Data Discovery, PostgreSQL is required.

3. The wizard checks your environment for needed prerequisites. If all items are all present, review the list and type `y` to continue.
4. If any prerequisites are missing:
   a. Type in `fixall` to fix all missing items, then select Enter.
   b. Type in `fixmandatory` to only fix the mandatory items required for your instance, then select Enter.
   c. Type in a number to fix a specific prerequisite, then select Enter to fix that item. Repeat as needed.

   Once all items are all present, review the list and type `y` to continue.

Step 3: Define Your Instance Name and Other Symphony Options

Define information about your instance by answering the appropriate prompts. To correct an earlier entry, type in `b` and select Enter to return to a previous prompt.

1. **Instance Name** work area: Provide an instance name for your Symphony installation and select Enter. Use only letters, numbers, and the underscore character.

   > **Important:** Windows support for Symphony includes one Symphony instance per Windows instance.

2. **Instance Description** work area: Provide an optional description for this instance and select Enter.

   > **Important:** Windows support for Symphony includes one Symphony instance per Windows instance.

3. **Instance Path** work area: Provide a value for the `InstancePath`, or select Enter to accept the default instance path.
4. **Use Empty Hostname** work area: Type in `Y` and Enter to use an empty hostname, or `n` to provide your own hostname information to use multiple sites on the same port.
5. **Enter Binding Details** work area: Select a binding type.
   a. Select `https` if you have appropriate certificates available. You'll be prompted to select an available certificate.
   b. Alternatively, select `http` as needed to work without encrypted communication, or to come back and set up as `https` later.

6. **Website Port** work area: Type in a port value, then select Enter. Alternatively, select Enter to accept the default port.
7. **Web Application Name** work area: Type in a new name and select Enter. Alternatively, accept the default defined by your earlier selections, and select Enter to continue.
8. **Application Pool** work area: Select, by number, the `NetworkService` options and select Enter to continue. Other options may include, depending on your environment: `SpecificUser`, or `ApplicationPoolIdentity`.
9. **Rabbit MQ Configuration** work area: Select, by number, either of these options, and complete the associated tasks to complete this step.
   a. **I have an existing RabbitMQ server and will provide its details.** You will need to provided the appropriate details to connect Symphony to your instance. Provide values for the **Rabbit MQ Host** (or accept the default), **Rabbit MQ Port** (or accept the default), **Rabbit MQ Username**, **Rabbit MQ Password**, and **Rabbit MQ Virtual Host**.
   b. **I need to install a RabbitMQ server.** Symphony walks you through installing Rabbit MQ and its Erland OTP dependency. You will need to provide values for the Rabbit MQ Username, Rabbit MQ Password, and Rabbit MQ Virtual Host. Keep note of that user name and the password you assign for future updates, upgrades, or changes.

> **Important:** If you do not install or connect Rabbit MQ at this time, you will encounter integration errors when you attempt to use both Managed and Discovery modules in Symphony.

Step 4: Define Symphony and Managed Dashboards and Reports Module Databases

After Rabbit MQ Configuration is complete (or skipped), you are prompted to provide details to configure databases by linking existing databases or installing new databases.

1. **Managed Overview** work area: If you have selected to install and license Managed Dashboards and Reports, enter `Y` to continue installation.
2. **Database State** work area: Define a database server for your instance. Have Symphony install PostgreSQL for you, or select an existing MSSQL instance, PostgreSQL instance, or . Select, by number, either of these options:
   a. **New**: Install a new database server. Provide a password and confirmation to define a PostgreSQL database.
   b. **Existing**: Link an existing database. Provide the details of your MSSQL or PostgreSQL database. This may include the Host Name, User Name, Password, and Database Name. SSL information may be required as well.
3. If you need to install a new database server, Symphony installs a PostgreSQL server for you. Provide a password and verify the password. The user name provided by Symphony is `postgres`. Keep note of that user name and the password you assign for future updates, upgrades, or changes.
4. If you select **Existing** to link an existing database, enter `Y` to provide database information for this instance.

a. **Application Storage Engine** work area: Select **SqlServer** or **Postgres**.

   For SQLServer, provide values for the **Data Source**, **User Id**, **Password**, and **Database Name** (or use default provided).

   For PostgreSQL, provide values for **Host**, **Port** (or use the default provided), **Username**, **Password**, **SSL Mode** option (**Prefer** is useful for most environments), and **Database Name** (or use the default provided).

b. **Warehouse Database Connection** work area, use the same database connection details you just defined by entering `y`, then select Enter. Accept or change the default Database Name value.

   Alternatively, enter `n`, select Enter, then provide details to connect to another database.

After this is complete, you are prompted to provide details to configure the Discovery module.

Step 5: Define Visual Data Discovery Module Settings

After Managed Configuration is complete, you are prompted to provide details to configure the Discovery module.

Note: These options are not provided if you are not installing the Visual Data Discovery module.

1. **Discovery Overview** work area: If you have selected to install and license Visual Data Discovery, enter `Y` to continue installation.
2. **Discovery Metadata State** work area: Define a database server for your instance. Have Symphony install PostgreSQL for you, or select an existing MSSQL instance, PostgreSQL instance, or select, by number, either of these options:

   a. **New**: Install a new database server. Provide a value for the **Discovery Postgres Server** or accept the default `localhost`. Type in a value for the **Port**, or use the default provided. Enter a value for **Discovery Postgres Username** and **Discovery Postgres Password**. Select an **SSL Mode** option (**Prefer** is useful for most environments), and all of the **Data Discovery** database names.

   Important: For a clean, uncomplicated instance of Symphony, we suggest you accept the default names provided by the installer.

   b. **Existing**: Link an existing database. Provide the details of your PostgreSQL database. This may include the Host Name, User Name, Password, and Database Name. SSL information may be required as well.

3. **Select Additional Features** work area: Enter numbers for available features to toggle them on or off for installation, then enter `Y` to continue installation.
4. **Select Discovery Connectors** work area: The PostgreSQL connector is selected by default. Enter a number for available connectors to toggle them on or off for installation, then enter `Y` to continue installation.

Step 6: Define Administrator Credentials

After Discovery Configuration is complete, you are prompted to provide Admin details for Symphony.

1. Provide a value for the **Admin** password, then enter it again to verify the password.
2. Provide a value for the Admin email address. Symphony provides a summary of the installation selections you have made.
3. If you are satisfied with your selections, type in `y` and select Enter to proceed with the installation process.
4. Once the installation is complete, you can review the completed tasks and navigate to the installation log provided to review this information in more detail if needed.

   Note: Keep note of that user name and the password you assign for future updates, upgrades, or changes.

Silent Installation

Step 1: Download and Launch Installation Packages

1. Download and install .NET 8.0 Desktop Runtime or later if it is not already present.
2. Install PostgreSQL if not already present.
3. Download the Symphony setup package.
4. Launch the application as a Windows administrator. A Logi Symphony Setup work area opens.
5. Enter an installation path, or accept the default path provided.
6. Select **Install Setup Files** to install the files you need to install the deployment console wizard.

   Important: Symphony will not installs PostgreSQL database for you as part of the silent install process. Install before you begin.
7. Once complete, you can close the Logi Symphony Setup work area and launch the Logi Symphony Deployment Console in one of two ways.

a. Open it from the Windows Start Menu by selecting **Logi Symphony Deployment Console**.

b. Launch it from the command line from its location, for example: `C:\Program Files (x86)\InsightSoftware\Logi Symphony\Setup\<version number>`.

Step 2: Generate and customize the deployment config file

1. Launch the Logi Symphony Deployment Console.
2. Type in the command `Logi.Symphony.Deployment.Console.exe` and select Enter. Several options for installing and managing your instance appear.
   a. `{Deployment Config File Path}` - Provide the path to your customized deployment config file to silently install Symphony.
   b. `-- ExportDeploymentConfig {Optional Path}` - Exports examples of all possible configs to a directory. Use to create a config file you can customize to create your silent installation.
   c. `-- wizard` - Launch the **Logi Symphony Deployment Wizard - Main Menu**.
   d. `-- Version` - Find out the current version of your software installer.
3. Generate a the sample config file, and supporting files. These include:
   a. `AddLogiSymphonyInstanceDeploymentConfig.xml` - Customize with your environment specifics to install in your instance. See Installation Steps - All Modules for more details on each item included.

      Important: You must ensure all prerequisites for installation are met, or the installation will fail. Fix your prerequisites by editing and running a customized `FixPrerequisiteDeploymentConfig.xml` in your environment.

   b. `FixPrerequisiteDeploymentConfig.xml` - Use to customize and install all prerequisite items in your environment.
   c. `RemoveLogiSymphonyInstanceDeploymentConfig.xml` - Customize and use to remove Symphony from your environment.
   d. `UpgradeLogiSymphonyInstanceDeployentConfig.xml` - Customize and upgrade your instance to upgrade to the version you've downloaded the files for.
4. After you've edited the appropriate files, call `Logi.Symphony.Deployment.Console.exe` and the path to the file you want to pass. Repeat for all needed files until installation is complete.

5. Once installation is complete, you can change the default settings for all services, as well as configure Visual Data Discovery services and Managed Dashboards and Reports services as needed.

6. Perform any post install options, as needed.

Post Installation Options

After you've completed installation, you can perform many of the tasks available in the Deployment Wizard, such as:

- **Logi Symphony - View**: Select to open your instance and begin adding users, data sources, and more.

  Note: If you need to set or reset your admin password, see How to Reset the Admin Password.

- **Managed - Add Federated Authentication To Instance**: Select to add a federated authentication option for your instance.
  - Select an instance from available options
  - Enter a value for the **Federated Authentication Application Pool** name, or accept the default name
  - Select an Identity Option (**NetworkService** is useful for most environments)
  - Confirm and add the service

- **Managed - Add GatewayHub to Instance**: Select to add the GatewayHub to your instance.
  - Select an instance from available options
  - Confirm and add the service

- **Discovery - Update Connectors and Additional Components**: Select to add or remove data connectors or other Discovery components.
  - Select an instance from available options
  - Enter a number for a feature to toggle for deployment or removal, then `Y` to continue
  - Enter a number for a data connector to toggle for deployment or removal, then `Y` to continue
  - Confirm and deploy the changes

For information about accessing Symphony after it is installed, see Access ComposerSymphony.

Installation Options Symphony - Windows

The Symphony Installation Wizard is a flexible tool you can use to install, upgrade, and manage your Symphony instance in a Windows environment.

Wizard Options

The Logi Symphony Deployment Wizard - Main Menu includes several install, upgrade, and update options. Make your selection by selecting the appropriate number showing in the work area. Options include:

- Logi Symphony - Add Instance. Add an instance of Symphony to your environment.
- Logi Symphony - Add Discovery to an Existing Instance. Use if you've installed Symphony with Managed only and want to add Data Discovery.
- Logi Symphony - Upgrade Instance. Use to upgrade to the latest major or minor version.
- Logi Symphony - View. Open your existing instance.
- Logi Symphony - Remove Instance. Use to uninstall Symphony. This does not remove the setup files and may not remove all databases.

  Note: After you have uninstalled Symphony from Windows, use the appropriate Add/Remove tool in Windows to uninstall the setup tools.
- Managed - Add Federated Authentication Module to Instance.
- Managed - Remove Federated Authentication Module from Instance.
- Managed - Add Sample. Use to add sample projects to Symphony.
- Managed - Add GatewayHub to Instance.
- Managed - Remove GatewayHub from Instance.
- Discovery - Update Connectors and Additional Components. Use to add or remove connectors or other components.

System Requirements

Symphony can be downloaded and installed in a Kubernetes environment.

Kubernetes Prerequisites and Required Resources

The following prerequisites are required for a successful deployment using Kubernetes and the Helm chart for Symphony:

1. A Kubernetes cluster of appropriate compute availability.

i. Install Kubernetes or have access to a cluster.

ii. The Helm CLI installed.

iii. A local copy of `kubectl` configured to work with your cluster.

2. Helm (3.8-3.11). See Installing Helm.

3. Access to Docker Hub.

The SymphonyHelm chart was developed with portability in mind. It relies only on Vanilla Kubernetes features and doesn't use any (cloud) vendor-specific features. Testing is performed in an Amazon EKS cluster and Minikube: you should need to make little to no modifications while working with other flavors of Kubernetes such as Azure AKS or Google GKE.

Required Resources - Managed Dashboards and Reports

There are no specific limits in the Helm chart by default, but we suggest the following compute availability on your cluster:

Recommended Compute Availability

- 10 Cores
- 24 Gi Memory

Required for Scheduling by Kubernetes

- 4 Cores
- 16 Gi Memory

> **Important:** These specifications are intended as an initial installation guide. Your environment's specific workloads may demand a higher number of cores and more memory available for computing.

Other Requirements

The memory requirements above do not take into account other components that are a part of Symphony. These components are external to Helm chart requirements. Size your environment following the recommendations for each component. These components include:

- PostgreSQL
- RabbitMQ
- NGINX Ingress Controller

Migrate from Composer to Symphony

This applies to: *Visual Data Discovery*

Whether you run Composer as a stand alone product or in an embedded environment, you can migrate supported versions to Logi Symphony following the steps outlined here. Generally, you'll create your intended (target) environment, back up your current (source) environment, move your data, install Symphony with a customized YAML, bring over your users, groups, and tenants, and adjust other specifics to your software environment.

- Migrate from Composer
- Typical Scenarios
- Baseline Migration Tasks
- Migrate an Embedded Environment
- Simple Embedding and Object-Level APIs
- Advanced User Management
- Authentication Changes
- Post Migration Steps

Important: If you are upgrading and have created an attribute named `User.timeZone`, this may be overwritten on upgrade. See Upgrade Workflow for more information about preparing your environment for the upgrade process.

Supported Versions

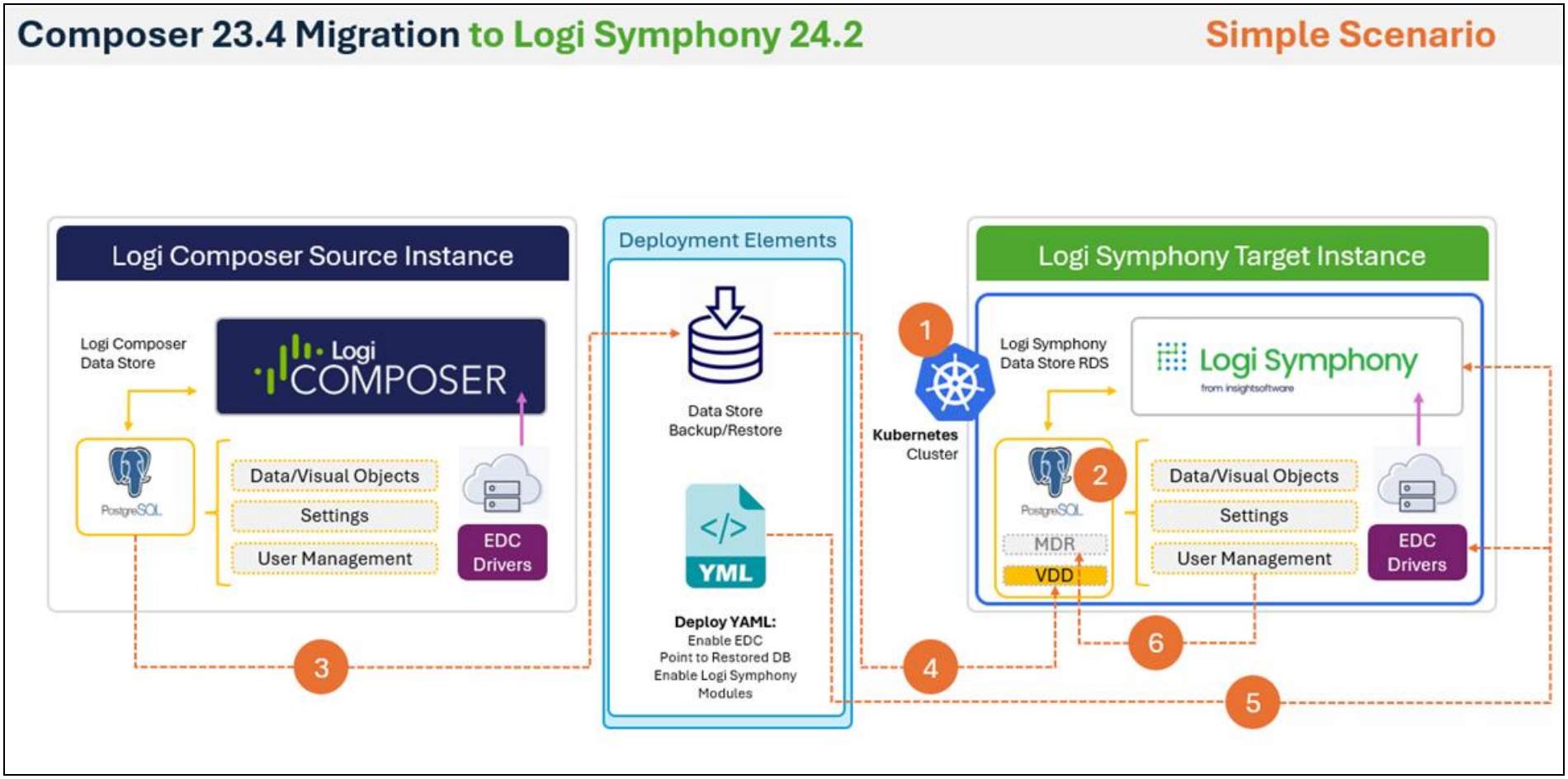
Logi Composer	Logi Symphony
23.4 and later	24.2 and later

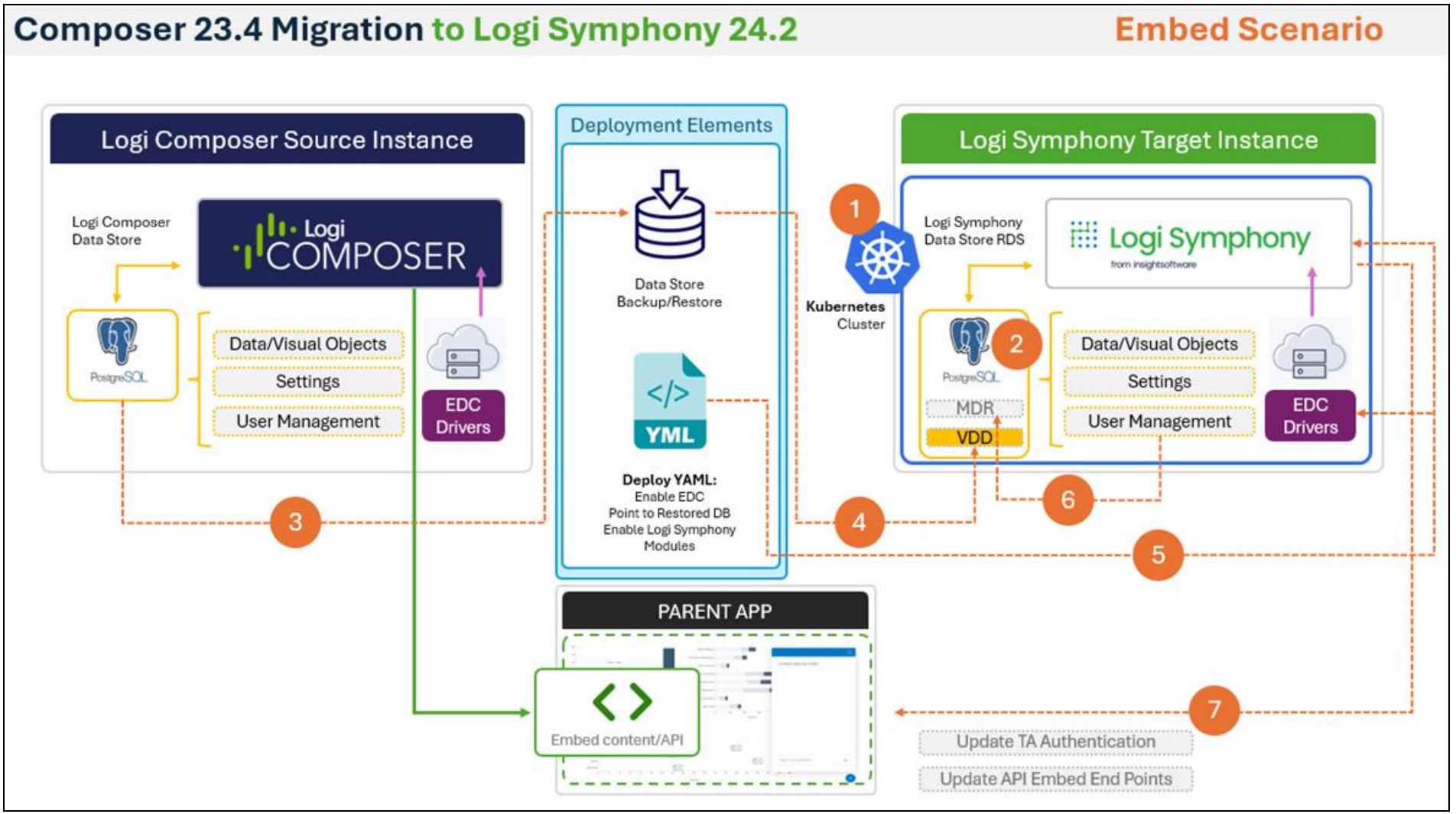
Migrate from Composer

Migrate your Composer stand alone or embedded environment by completing the tasks in this article. If you managed your users using the Composer / Visual Data Discovery API or support an embedded environment, there are further tasks to complete your migration.

Review the steps below, then prepare your target and source environments for migration, installation, and customization.

Typical Scenarios





Baseline Migration Tasks

1. Kubernetes Cluster

Before you begin, create your intended (target) environment to meet the following prerequisites, which are the baseline Symphony requirements. You will need:

1. A Kubernetes cluster (versions 1.23-1.27) of appropriate compute availability.
    a. Install Kubernetes or have access to a cluster.
    b. Install the Helm CLI.
    c. Include a local copy of `kubectl` configured to work with your cluster.
2. Helm (3.8-3.11). See Installing Helm.
3. Access to the insightsoftware Docker Hub at https://hub.docker.com/u/insightsoftware.
4. PostgreSQL 16 and above.
5. RabbitMQ.
6. NGINX Ingress Controller.

Prepare your environment, but do not install Symphony at this step.

2. PostgreSQL RDS

Install a supported version of PostgreSQL for your new Symphony instance.

3. Backup PostgreSQL from Your Source Instance

Back up your existing Composer databases (source databases).

Corresponding URL	Database Name and URL
`dbUrl`	`"jdbc:postgresql://10.0.0.0:5432/zoomdata-qe"`
`metadataDbUrl`	`"jdbc:postgresql://10.0.0.0:5432/zoomdata"`
`uploadDbUrl`	`"jdbc:postgresql://10.0.0.0:5432/zoomdata-upload"`
`keysetDbUrl`	`"jdbc:postgresql://10.0.0.0:5432/zoomdata-keyset"`
`userAuditingDbUrl`	`"jdbc:postgresql://10.0.0.0:5432/zoomdata-user-auditing"`

4. Create Databases and Restore PostgreSQL to Your Target Instance

Create the databases you'll need in Symphony for the modules you'll deploy

- Visual Data Discovery - create and restore your Composer data here.
- Managed Dashboards and Reports - create corresponding databases to support content shared among all installed components.
- Logi AI - create an AI database to support the use of Logi AI if deployed in your environment.

Visual Data Discovery

Create the following PostgreSQL databases in your target Symphony instance:

```
CREATE DATABASE "zoomdata-qe" OWNER "<defineuser>"
CREATE DATABASE "zoomdata" OWNER "<defineuser>"
CREATE DATABASE "zoomdata-upload" OWNER "<defineuser>"
CREATE DATABASE "zoomdata-keyset" OWNER "<defineuser>"
CREATE DATABASE "zoomdata-user-auditing" OWNER "<defineuser>"
```

Restore your backups from your Composer source to these target databases in Visual Data Discovery only.

Managed Dashboards and Reports

Create empty databases to support your Symphony environment. `<defineuser>` must match the owner you defined for the Data Discovery databases. Additionally, add the following extensions to the databases: `uuid-ossp`, `unaccent`.

```
CREATE DATABASE "dundasbi" WITH OWNER <defineuser>;
\connect "dundasbi";
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
CREATE EXTENSION IF NOT EXISTS "unaccent";
CREATE DATABASE "dundasbiwh" WITH OWNER <defineuser>;
\connect "dundasbiwh";
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
CREATE EXTENSION IF NOT EXISTS "unaccent";
```

Logi AI

If you're deploying Logi AI in your environment, create an empty AI database to support it in your environment. `<defineuser>` must match the owner you defined for the Data Discovery and Managed Dashboards and Reports databases. Additionally, add the following extension to the database: `uuid-ossp`.

```
CREATE DATABASE "LogiSymphonyAI" WITH OWNER <defineuser>;
\connect "LogiSymphonyAI";
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
```

5. YAML Updates for Logi Symphony Deployment

Start with the standard YAML provided as part of the Symphony deployment package for your target instance. You'll need to make a few important changes to the `global`, `managed`, `discovery`, and `logi` sections of the YAML. Include settings for `ai` if you are going to deploy Logi AI in your environment.

```
global:
  # Disable the PostgreSQL from the helm chart.
  postgresql:
    enabled: false

managed:
  dundas:
    bi:
      appDb:
        # Setup the Dundas BI Databases
        appDbConnString: "Host=someservername;Username=test;Password=test;Database=dundasbi"
        warehouseDbConnString: "Host=someservername;Username=test;Password=test;Database=dundasbiwh"
        appStorage: "Postgres"

discovery:

  # The database username and password.
  defaultDbUsername: "test"
  defaultDbPassword: "test"


  # The database urls.
  queryEngine:
    dbUrl: "jdbc:postgresql://someservername/zoomdata-qe"
  zoomdataWeb:
    metadataDbUrl: "jdbc:postgresql://someservername/zoomdata"
    uploadDbUrl: "jdbc:postgresql://someservername/zoomdata-upload"
    keysetDbUrl: "jdbc:postgresql://someservername/zoomdata-keyset"
    userAuditingDbUrl: "jdbc:postgresql://someservername/zoomdata-user-auditing"


logi:
```

```
symphony:
  ai:
    # This is assuming AI (BETA) is enabled.
    database:
      host: "someservername"
      name: "LogiSymphonyAI"
      auth:
        username: "test"
        password: "test"
```

1. Set `postgresql` to `enabled:false` (the default is `enabled:true`)

```
postgresql:
  # Enables a shared PostgreSQL server. This is defaulted to true,
  # however it is recommended in production to use an external database
  # See
    enabled: false
```

2. Change the database user name and password to match those you used during database creation (step 3) and restoration (step 4).

```
discovery:
  postgresql:
    # Do not change, when using Logi Symphony, the Postgres should come from the parent Logi Symphony chart.
    enabled: false

    # Do not change, when using Logi Symphony the ingress should come from the parent Logi Symphony chart.
    ingress:
      installController: false
      enabled: false

    defaultDbUsername: YourDatabaseUser
    defaultDBPassword: YourDatabasePassword
```

3. Point your databases to the correct URL, IP, port, and database names.

```
query engine:
  dbUrl: "jdbc:postgresql://10.0.0.0:5432/zoomdata-qe"

    resources:
    limits: {}
    requests:
      cpu: 500m
      memory: 1.5Gi

zoomdataWeb:
  metadataDbUrl: "jdbc:postgresql://10.0.0.0.5432/zoomdata"
  uploadDbUrl: "jdbc:postgresql://10.0.0.0.5432/zoomdata-upload"
  keysetDbUrl: "jdbc:postgresql://10.0.0.0.5432/zoomdata-keyset"
  userAuditingDbUrl: "jdbc:postgresql://10.0.0.0.5432/zoomdata-user-auditing"
  resources:
```

4. Save and deploy your revised YAML to your target environment.

6. Use the API to sync your imported users from Composer to the Symphony Instance

At this point, you can log in to Symphony and see your content in the Visual Data Discovery module. No users, groups, or tenants are present in your overall environment. Use a POST API tool of your choice to sync the users to the full environment.

Run the synchronization API

1. Determine your Logon Session ID. Use this in your API tool to perform sync tasks.

2. Define the licenseSeatKind in the body of your API POST. For example:

```
{
       "licenseSeatKind": "4"
}
```

licenseSeatKind	Value
Reserved Developer	0
Reserved Power User	1
Reserved Standard User	2

licenseSeatKind	Value
Floating Developer	3
Floating Power User	4
Floating Standard User	5

3. POST to this endpoint: `{{baseUrl}}/api/Admin/MigrateAccountServicesObjectsFromDataDiscovery?sessionId=<sessionId>`
4. Execute to synchronize all users across Symphony.

Migrate an Embedded Environment

If you run an embedded Composer environment, perform the tasks above to migrate your data, objects, and users.

Simple Embedding and Object-Level APIs

Once that is complete, make changes to your environment to enable your users to access objects.

- Change the URL to include `/discovery` in the API paths. From `/api/<endpoints>` to `/discovery/api/<endpoints>`.
- Update the Authentication TA API for your new URL.
- Adjust the Embed Manager token URLs.
- Update any custom APIs in use. See the Playground for examples.

API documentation is provided with your Symphony installation at this link: *`<symphony-URL>/discovery/swagger-ui.html`*.

Advanced User Management

If you are coming from a Composer only environment, you'll need to create users, groups, and tenants using the Symphony user interface.

If you used the Composer / Visual Data Discovery API endpoints, to create users, groups and tenants, you will now need to use these Symphony endpoints to manage these tasks:

- **Users**: /Account/
- **Groups**: /Group/
- **Tenants**: /Tenant/

Note: Note: Some references in the linked documentation refers to User accounts as Accounts.

Authentication Changes

If you are using Trusted Access, you should update your endpoints to include `/discovery/` in appropriate paths.

If you intend to manage users, groups, and tenants using API calls, set up authentication following the information provided here: Create and Use Logon and Session Tokens for Symphony.

Post Migration Steps

1. Install the appropriate licenses for Symphony. See Add a License.

   Important: If you are upgrading to Symphony, Visual Data Discovery module from Composer, you will need add a license for Logi Symphony as well.

2. Update any endpoints you use that have changed by replacing `/composer/` with `/discovery/` where appropriate. This impacts:
   a. bookmarks your users may use
   b. federated authentication redirect URLs
   c. proxy servers
   d. API endpoints you may use

3. Inform users they'll need to use the "Forgot my password" option when they log in for the first time to your new instance.

   Important: If users don't have an email address associated with their user account, run a script to return a list of users who are missing an email address.

Post-Installation Options

This applies to: *Visual Data Discovery*

After the Symphony environment has been installed in your server, the following configuration options are available to you:

To	Read
Add an SSL Certificate. By default, Symphony enables an *http* port. To enable *https*, you must add an SSL certificate to the Symphony server.	Add an SSL Certificate
Disable SSL.	Disable the SSL Certificate in ComposerSymphony Data Discovery
Use SQL-based connectors. Some SQL connectors require a JDBC driver to be configured before you can connect to your data source. You can download the driver from the vendor's site. Be aware that you need to download and configure JDBC drivers for the following Symphony connectors as soon as you complete the Symphony installation: ■ MemSQL ■ MySQL ■ Oracle ■ Amazon Redshift ■ Teradata	Add A JDBC Driver
Configure Symphony memory settings.	Configure Memory Settings
Use Symphony's sample data generator.	Manage the Real Time Sales Demo Source

How to Enable SQL Server Authentication

This applies to: *Managed Dashboards, Managed Reports*

SQL Server authentication is recommended for connecting Symphony to its application and warehouse databases for security reasons. This article explains how to enable SQL Server authentication, and how to use it in your Symphony environment.

This applies when installing or upgrading Symphony when it uses Microsoft SQL Server to store its own data. Within Symphony, users can then connect, analyze, and visualize data from a variety of other data sources or use other authentication methods.

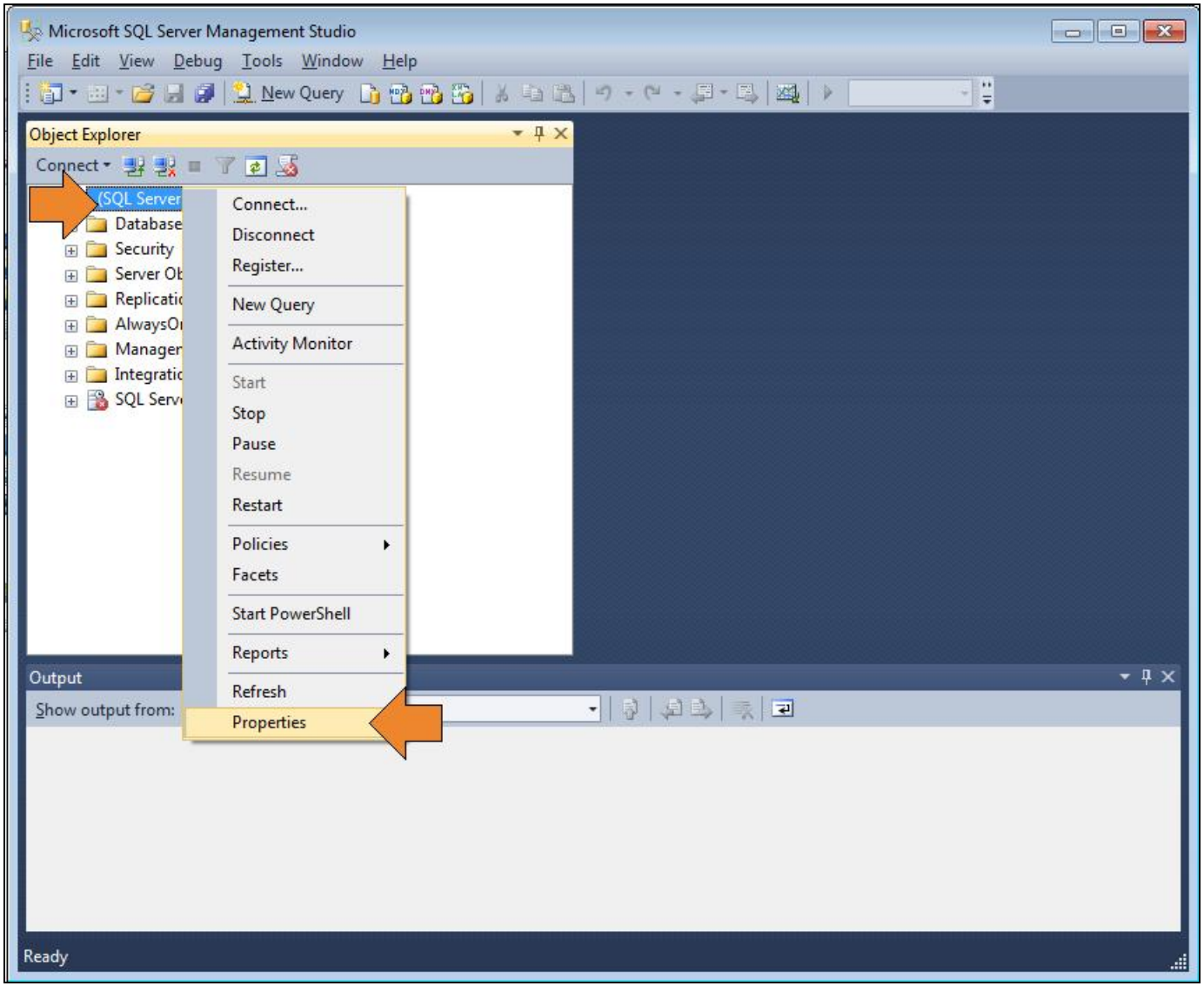
Note: You should have a basic understanding of using SQL Server Management Studio to deploy in your environment.

Enabling SQL Server Authentication through SQL Management Studio

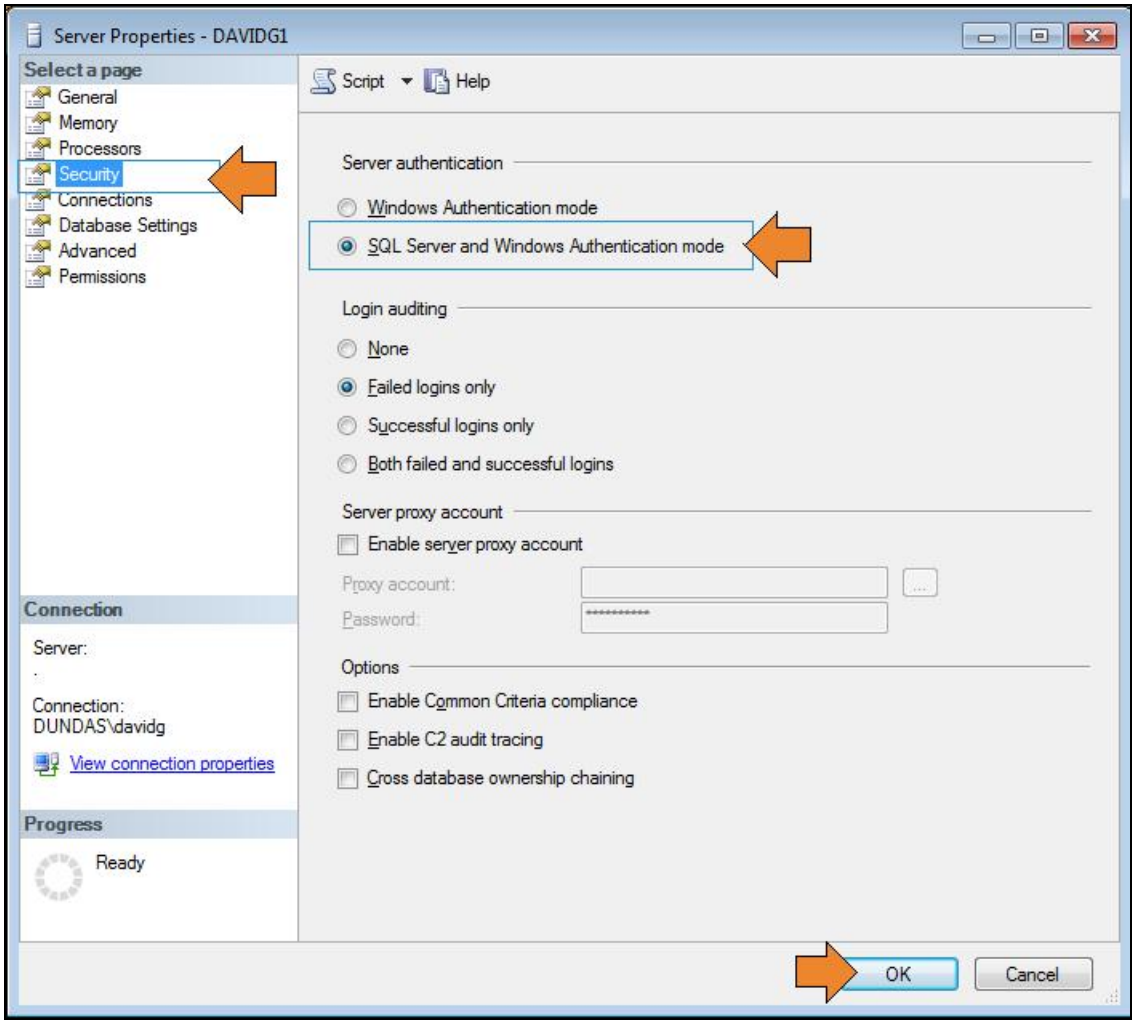
Enable SQL Server Authentication for your instance

1. Open SQL Server Management Studio.
2. Connect to the SQL Server instance you would like to use for Symphony.

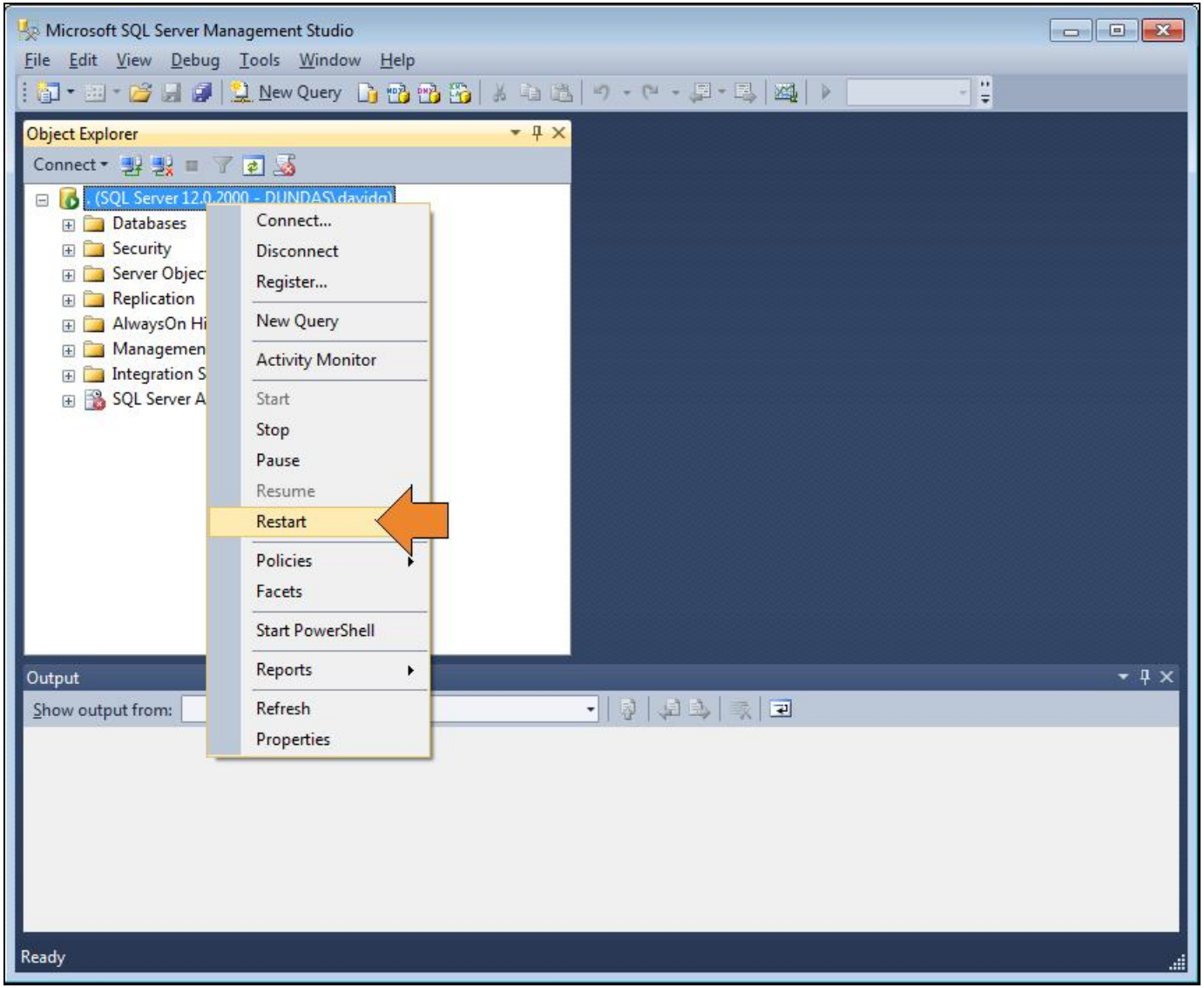
3. In the Object Explorer, right-click the server and click **Properties**.



4. On the **Security** page under **Server authentication**, select **SQL Server and Windows Authentication mode** and then click **OK**.



5. In the Object Explorer, right-click your server and click **Restart**. If the SQL Server Agent is running, it must also be restarted.



Using SQL Server Authentication in Symphony

Certain database permissions are needed for the user Symphony connects as while deploying a Symphony instance, which can be reduced when deploying is finished.

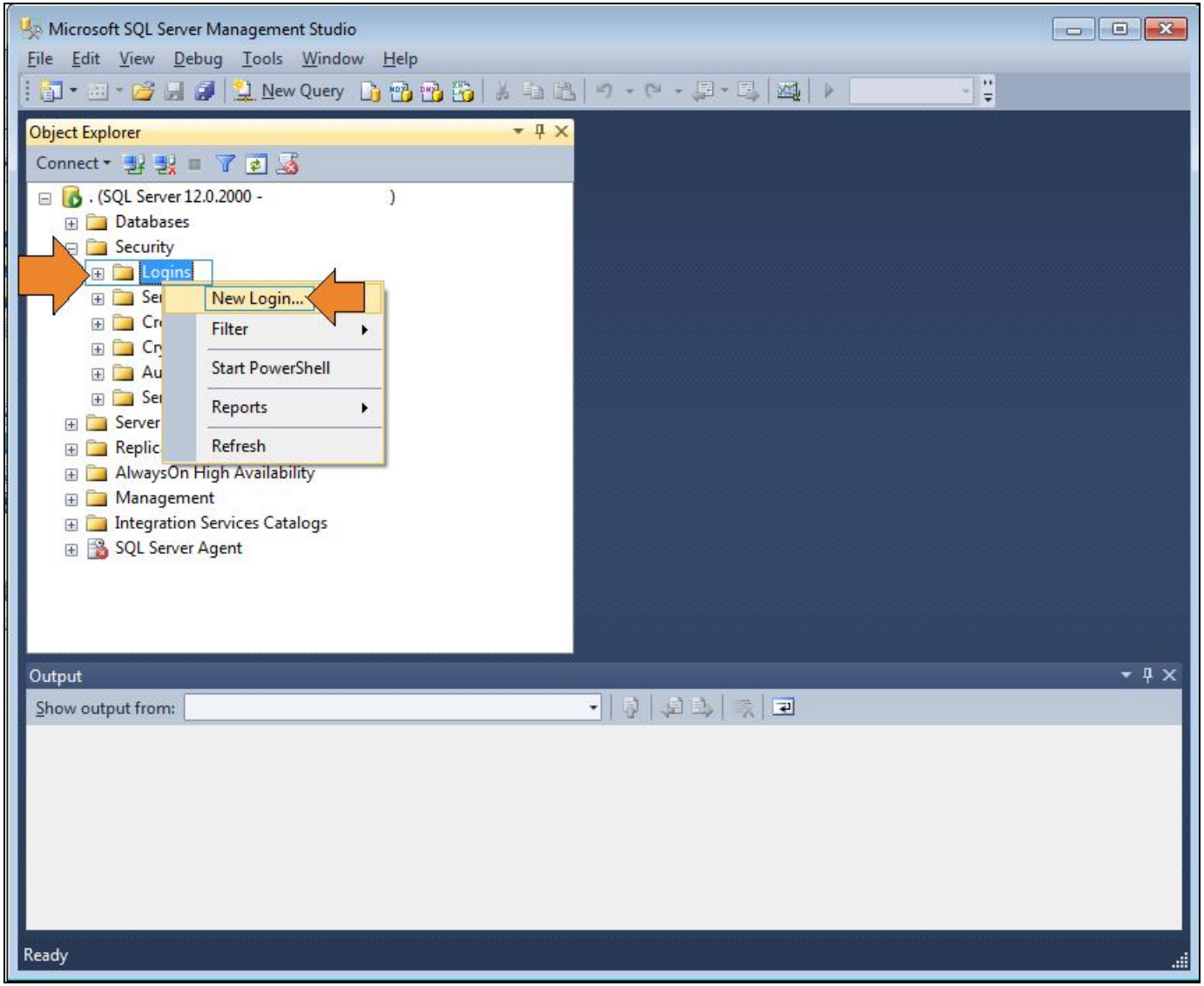
Before Deploying an Instance

Create a SQL Server authentication user

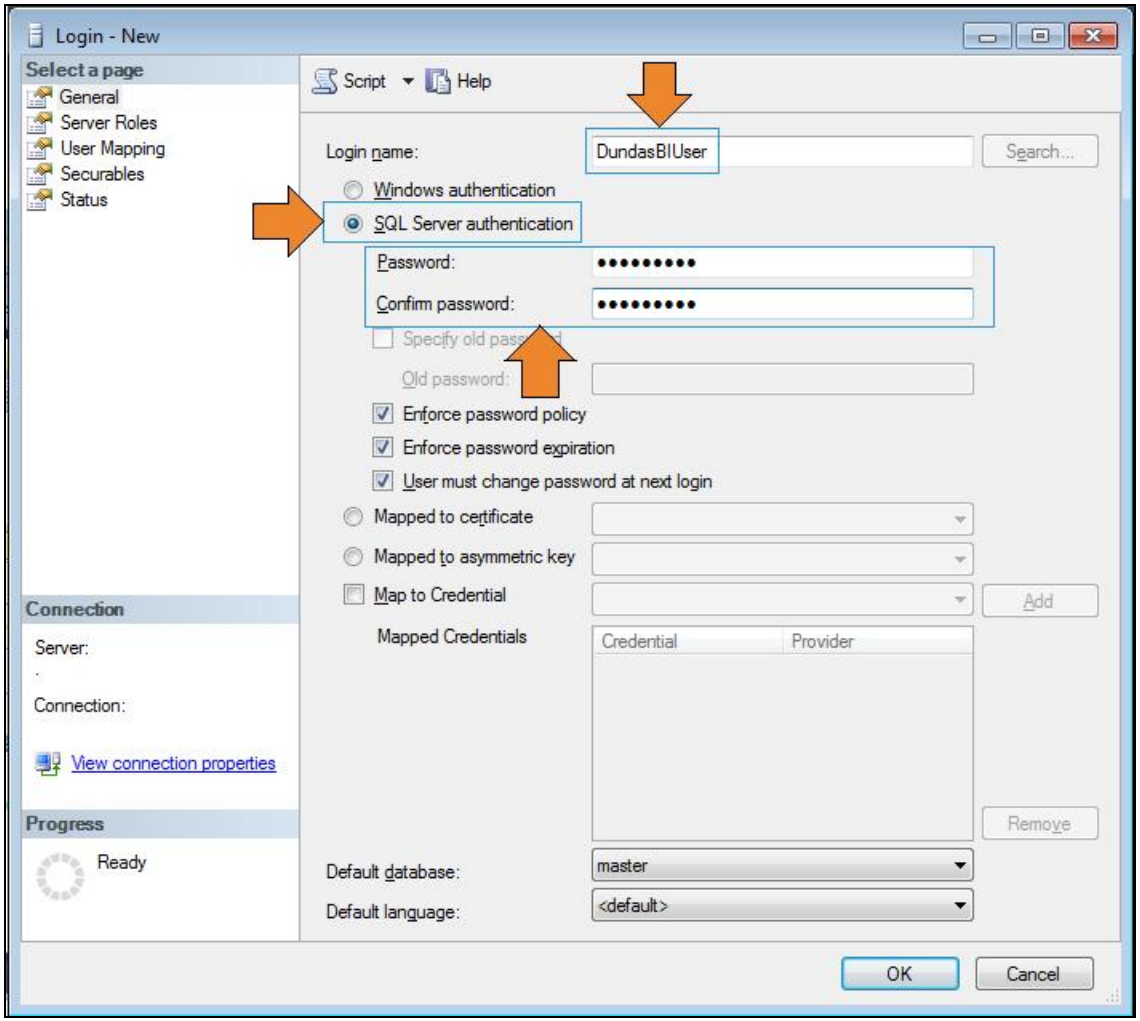
During the deployment of an instance with new databases, the user must have the **SysAdmin** role, or else all of the following: **DbCreator**, **DiskAdmin**, **ProcessAdmin**, and **SecurityAdmin**.

Create a user like this in SQL Management Studio:

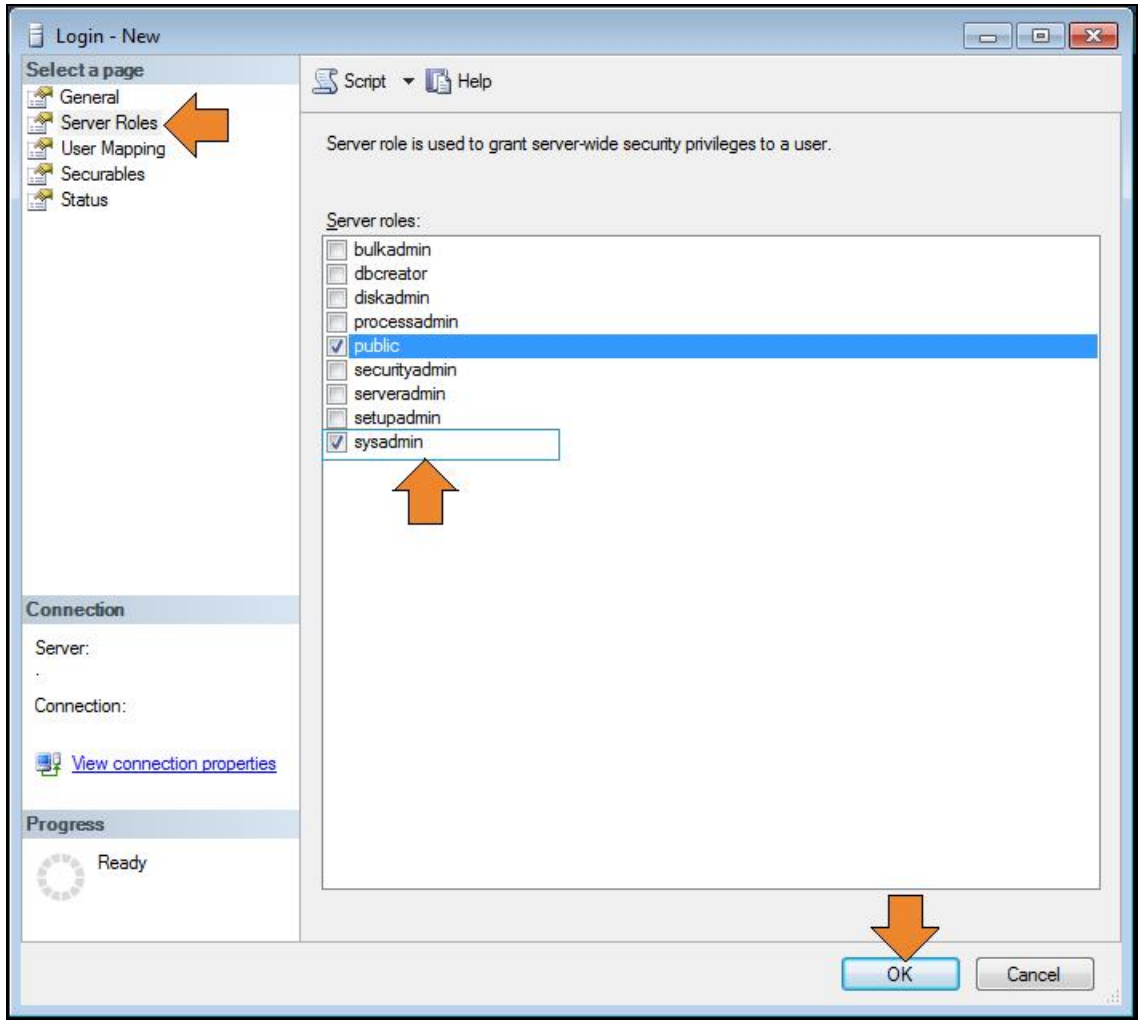
1. Open SQL Server Management Studio.
2. Connect to the SQL Server instance you would like to use for Symphony.


Microsoft SQL Server Management Studio
File Edit View Debug Tools Window Help
New Query
Object Explorer
Connect
. (SQL Server 12.0.2000 -)
Databases
Security
Logins
New Login...
Filter
Start PowerShell
Reports
Refresh
AlwaysOn High Availability
Management
Integration Services Catalogs
SQL Server Agent
Output
Show output from:
Ready

4. Enter the login name as **DundasBIUser**, select **SQL Server authentication**, and enter the Password.

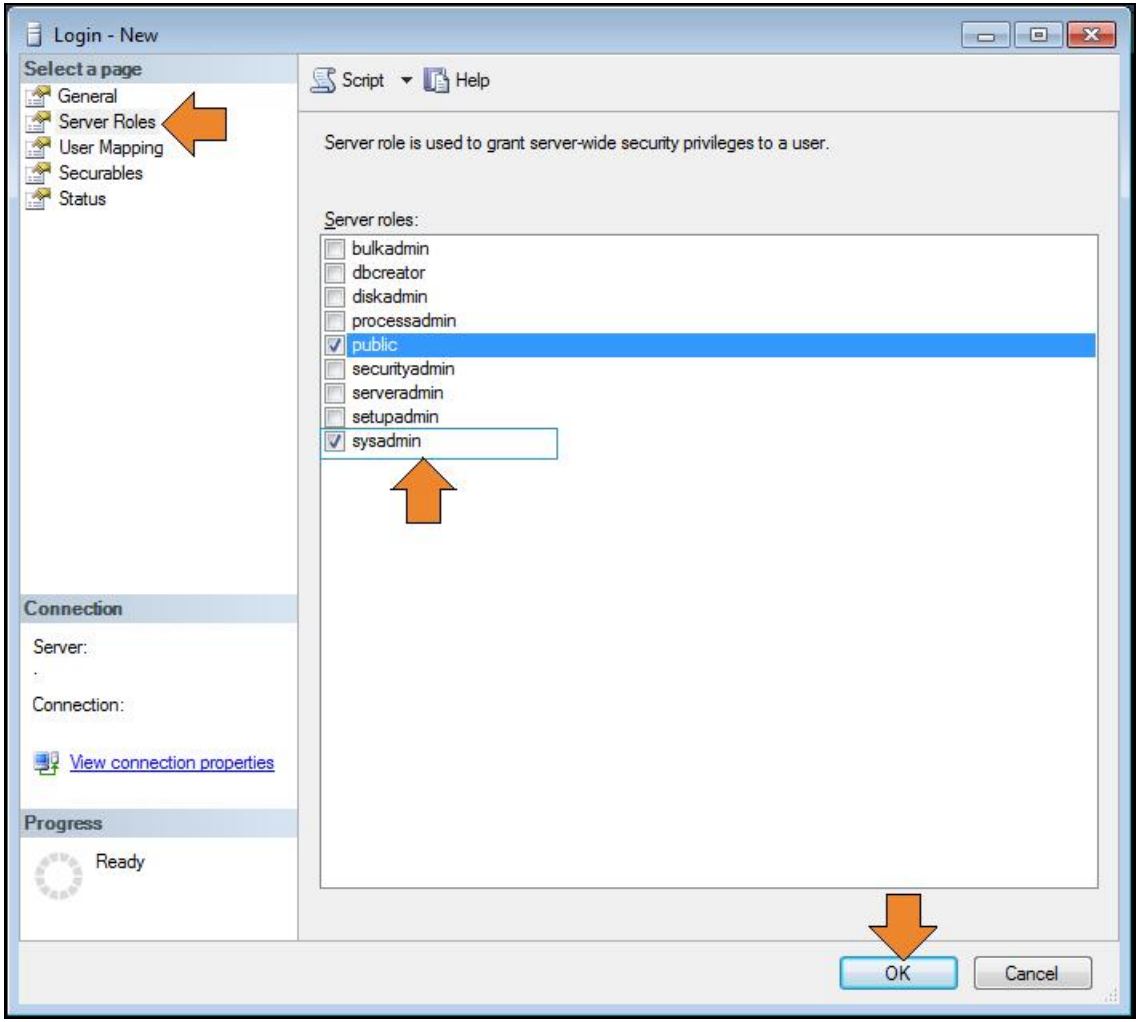


5. Click the **Server Roles** page and enable the **SysAdmin** role, or a combination of **DbCreator**, **DiskAdmin**, **ProcessAdmin**, and **SecurityAdmin** roles.



Specifying the SQL Server Authentication User

Now that the SQL Server authentication user has been created, use these credentials when you deploy Symphony and set up the application database connection.



After Deploying an Instance

After deploying, it is possible to remove the **SysAdmin** role from this user. For regular operation, the user will only require the **dbo** default schema and the **db_owner** role membership.

The **SysAdmin** role will be required again when attempting to upgrade an instance.

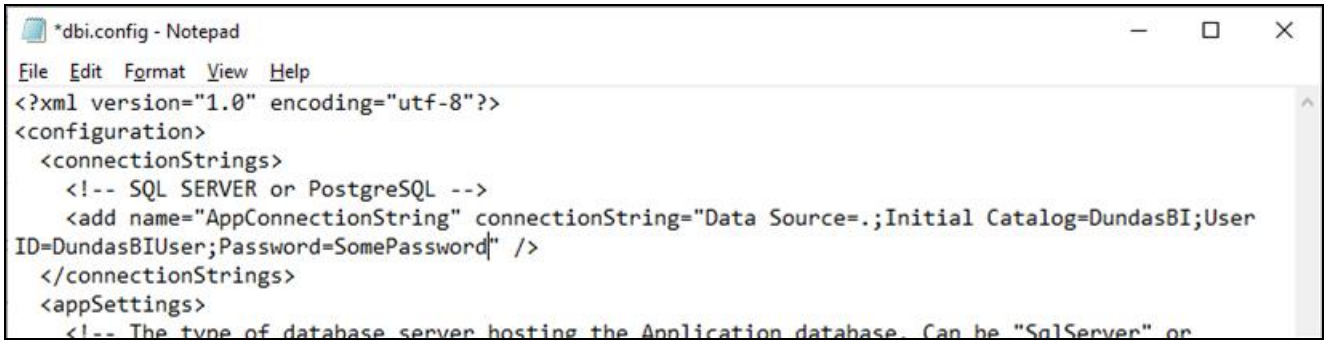
Change an existing Instance to Use SQL Server Authentication

Follow the steps below to change an existing instance to use SQL Server authentication for its application and warehouse databases.

Application Database

Open a text editor such as Notepad as an administrator, by right-clicking its shortcut and choosing **Run as administrator**. Open the configuration file.

Edit the connection string to use SQL Server authentication by specifying the User ID and Password. After saving, recycle the application pool in IIS or restart the Linux service for Symphony's website.



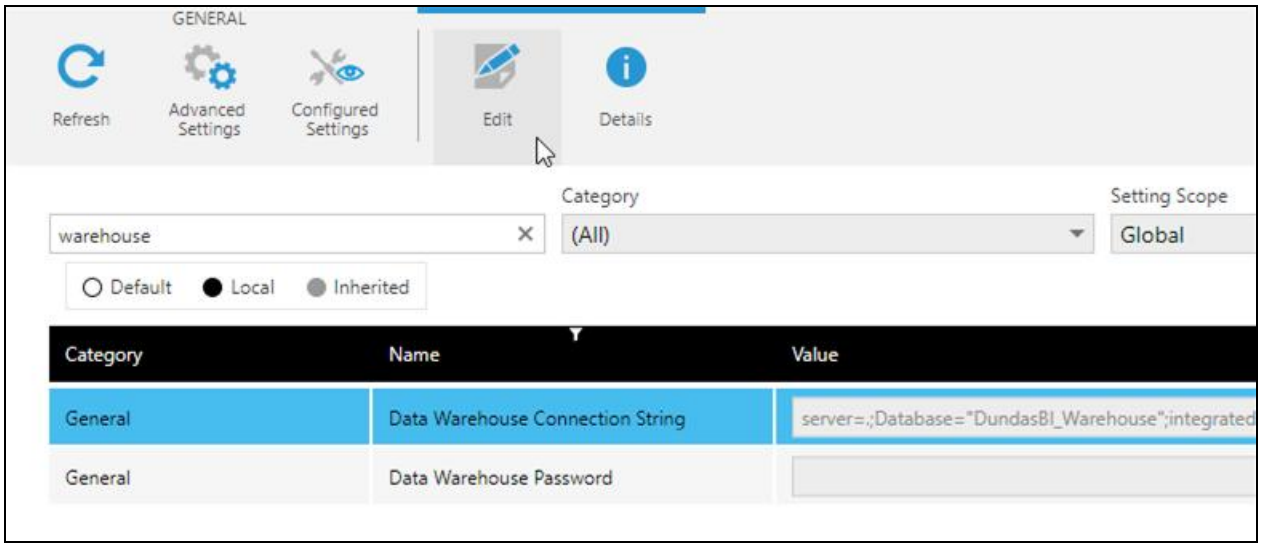
Note: To change the credentials when the connection string is encrypted, first decrypt the connection string using the dt command line tool, then update the credentials. To resume the secure state, encrypt the connection string when you have updated the credentials.

Warehouse Database

The warehouse database connection is defined in Symphony's configuration settings.

Log into Symphony as an administrator, and access Administration from the main menu.

Click to expand **Setup** and then click **Config**.



For more information, see:

- Microsoft Docs: SQL Server Management Studio
- Microsoft Docs: Choose an Authentication Mode
- Symphony Managed Dashboards and Reports Config File
- Configuration Settings

Install and Configure R

This applies to: *Managed Dashboards, Managed Reports*

R is a popular programming language and environment for statistical computing and predictive analysis.

Symphony integrates with an existing R system at the data cube level via two transforms which you can use to supply data to the cube or perform data processing/analysis on other data.

In order to use the R-related functionality in Symphony, you must have access to an existing R server. There are configuration settings in Symphony that allow administrators to specify the details for connecting to an external R server.

Install the R Server

The open source Rserve (Binary R Server) software is available for Unix, Linux, and Windows platforms, however the Windows version is not recommended other than for evaluation purposes due to limitations.

The Windows version of Rserve works in cooperative mode only, which means that only one connection at a time is allowed and all subsequent connections share the same namespace. For example, using both an R Data Generator transform and an R Language Analysis transform in the same data cube will result in an error because of this limitation.

Unix / Linux

For details on installing Rserve on Unix or Linux systems, see https://rforge.net/Rserve/doc.html#intro.

Windows

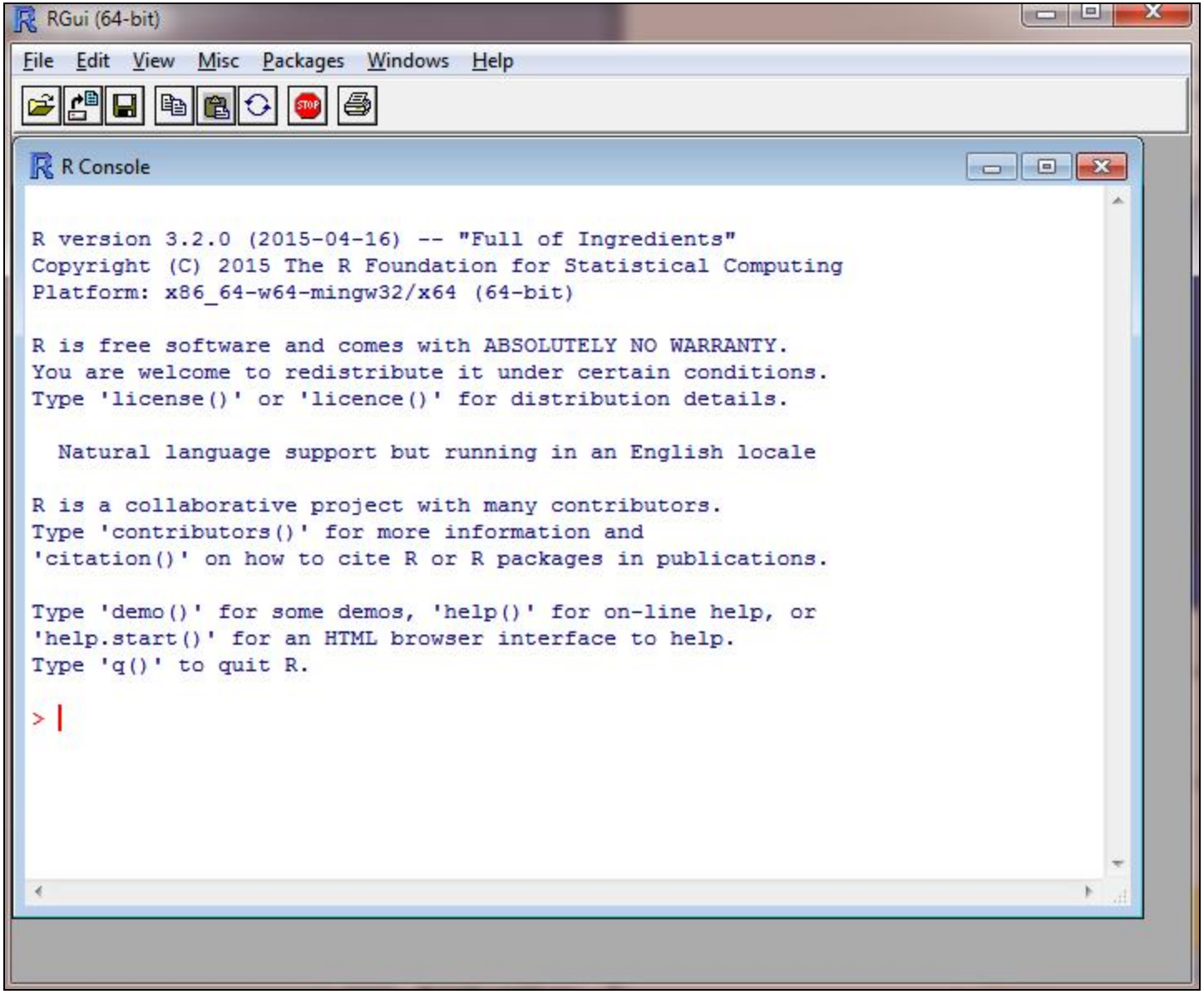
If you want to run Rserve on Windows with Symphony (for testing or non-production purposes), first download and install R for Windows in your environment.

This installer package will install the R binaries to a folder similar to: **C:\Program Files\R\R-3.2.0\bin\x64**

Once R for Windows is installed, add the above folder path to your Windows Path environment variable. This will allow the Rserve program to locate `R.dll`.

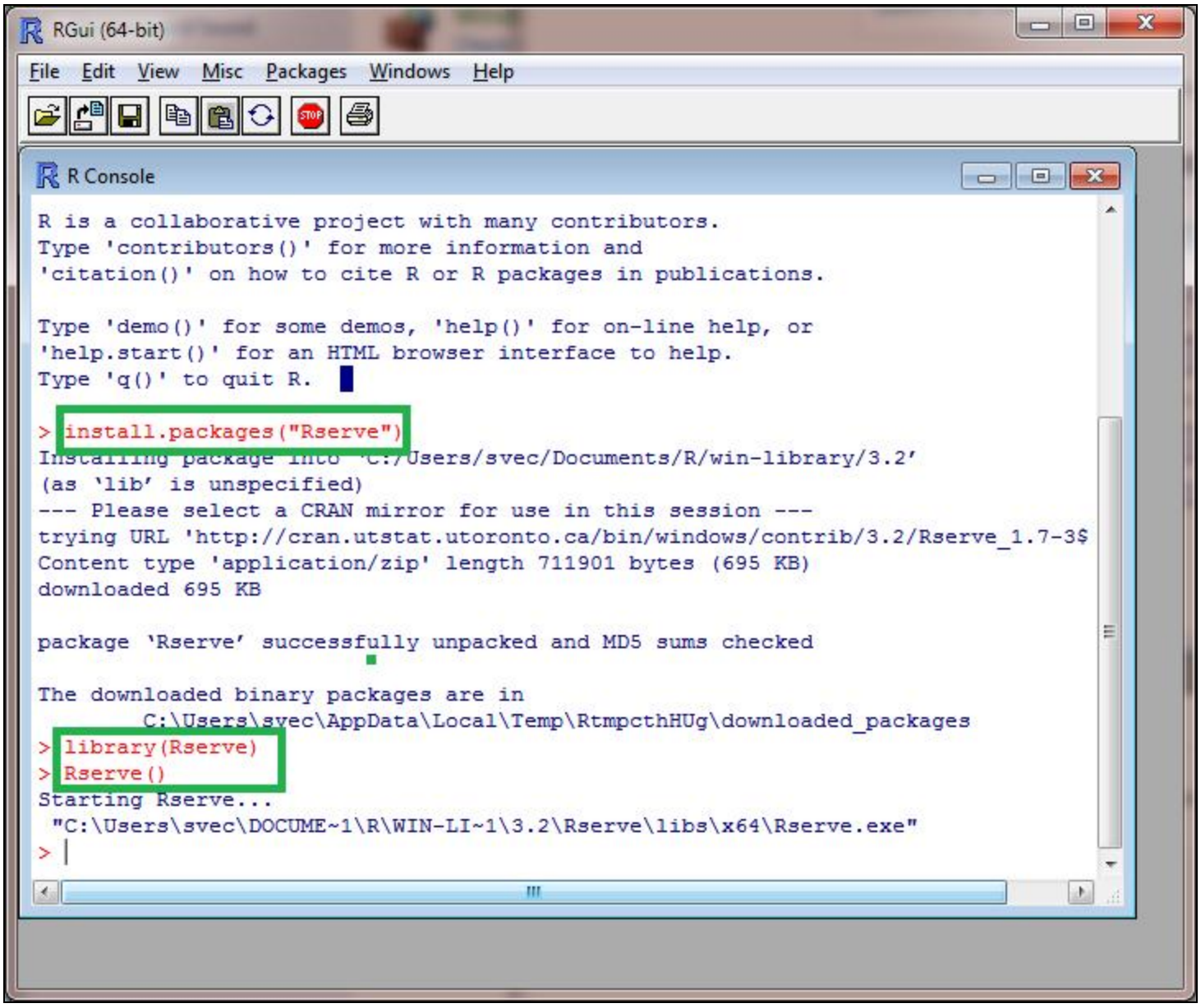
Next, go to your Start Programs menu and expand the newly installed R folder.

Click **R x64 3.2.0** (or similar) to launch the **R Console**.



In the R Console, type the following commands in order to download and install Rserve (this is the R server) and run the program.

```
install.packages("Rserve")
library(Rserve)
Rserve()
```



The R server is now running and you can use the R transforms in Symphony as described in the articles for the R Data Generator and R Language Analysis transforms.

Note: To run Rserve without the R Console, open a Command Prompt window and start the Rserve.exe program. You can also copy the Rserve.exe program to your Windows Startup folder to have it run automatically when Windows starts.

Configure Rserve Connection Settings

Once your R server is running, you'll need to update the Rserve Connection settings in Symphony in order to properly connect to the R server.

For more information, see:

- R Data Generator
- R Language Analysis
- Configuration Settings

Install Python

This applies to: *Managed Dashboards, Managed Reports*

Python is a popular programming language that can be used for advanced analytics and data modeling. Symphonyintegrates with Python at the data cube level via two transforms, which you can use to supply data to the data cube or perform data processing and analysis on other data.

Important: The minimum supported version of Python for Managed Dashboards and Reports is Python 3.8.

Install Python

To install Python on Windows, download and run the installer for the 64-bit version of Python.

On Linux, the standard default version of *python3* provided by supported Linux distributions and versions is normally used except where noted otherwise in the system requirements for your version of the software.

The process on Windows is generally:

1. Run the Python setup.
2. Select **Customize installation** and **click Next** on the **Optional Features** page.
3. Select **Install for all users** on the **Advanced Options** page and click **Install**.

Install Packages

Packages other than those installed automatically by Symphony must be installed by an administrator on the server that runs Symphony before they can be used in Python transforms.

For example, to install the common **pandas** package, which includes other packages such as NumPy:

1. Open the command prompt or terminal on the Symphony server as an administrator (e.g., right-click on Command Prompt in the start menu and choose **Run as administrator**).
2. You may need to navigate to the particular **Scripts** folder corresponding with the version of Python used by Symphony. For example:

```
cd C:\Program Files\Python38\Scripts
```

3. Use **pip** to install the package. For example:

```
pip install pandas
```

- The Python-related transforms in Symphony require a result to be returned by the Python script, which can be represented as a table. This includes results generated using the NumPy and Pandas packages for data analysis.
- You may be required to restart the server after installing Python.

For more information, see:

- Python Data Generator
- Python Analysis

Using A Managed Dashboard and Managed Reports Gateway

This applies to: *Managed Dashboards, Managed Reports*

You can use a gateway to connect to your on-premise data from an instance of Symphony.

Once the gateway is installed on your local network and configured in Symphony, users can create data connectors to access the following on-premise data sources via the gateway:

- dBase (DBF) files
- Flat files
- IBM Db2 (v24.3 and later)
- JDBC
- MemSQL
- Microsoft Excel
- Microsoft SQL Server
- MySQL
- OData
- ODBC (install ODBC drivers to use for additional data sources)
- Oracle database
- PostgreSQL
- Teradata
- XML files

Note: For the full list of data sources supported without a gateway, see the system requirements. You don't need a gateway for cloud or publicly accessible data sources, or for uploaded files.

System Requirements

Some data sources may require drivers to be installed on the same computer as the gateway. The gateway also has minimum requirements for the operating system and for .NET that are similar to the server requirements for a Symphony instance, but the minimums listed for evaluation use can also be suitable for a gateway.

Note: The gateway should be installed on a computer that remains online in order to prevent errors when using data connectors in Symphony that use this gateway.

Linux Installation

The Symphony Deployment wizard is used to create and manage gateways on Linux.

Follow the instructions in Installing Symphony on Linux for downloading and installing the Deployment wizard for your distribution of Linux, but do so on a computer that can be used to provide access to your data sources. Rather than launching the wizard to create an instance, you will create a gateway as shown below.

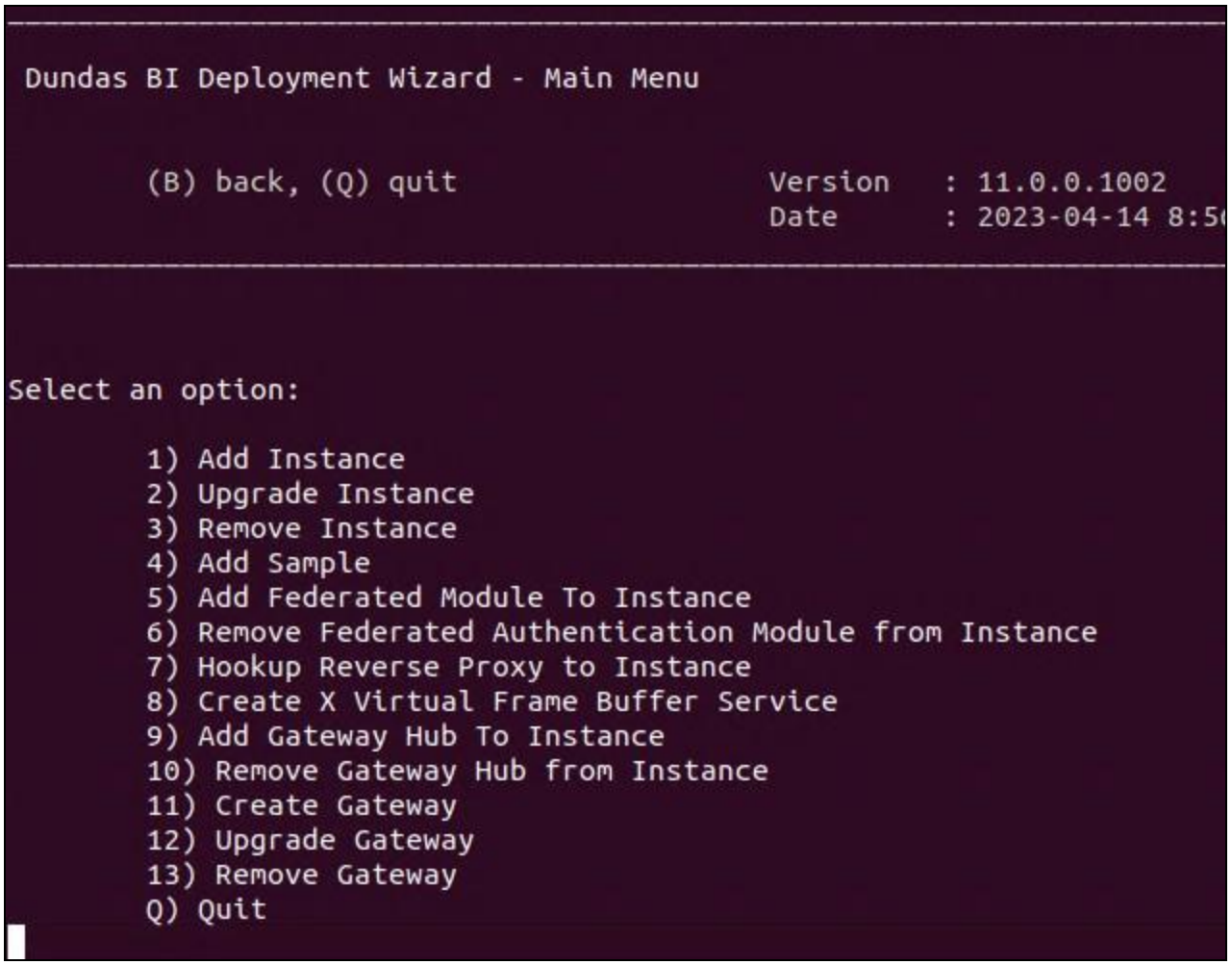
When a new version of Symphony becomes available, you can download and install the new version of the Deployment wizard, then run the step to upgrade your existing gateway.

Managing Gateways

Launch the Symphony Deployment wizard in a terminal using the --gateway option. For example:

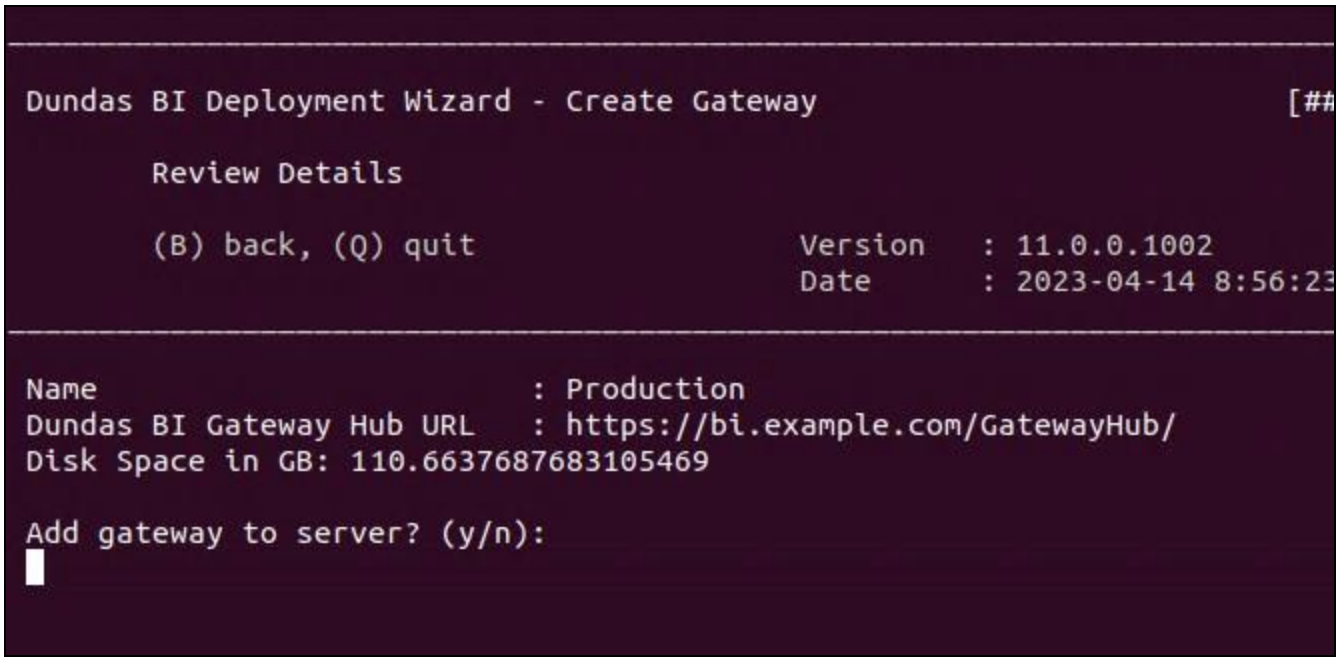
```
sudo dundasbi --wizard --gateway
```

To create a new gateway, type the number indicated for the step **Create Gateway** followed by the enter key. There are also steps available to **Upgrade Gateway** and **Remove Gateway**.

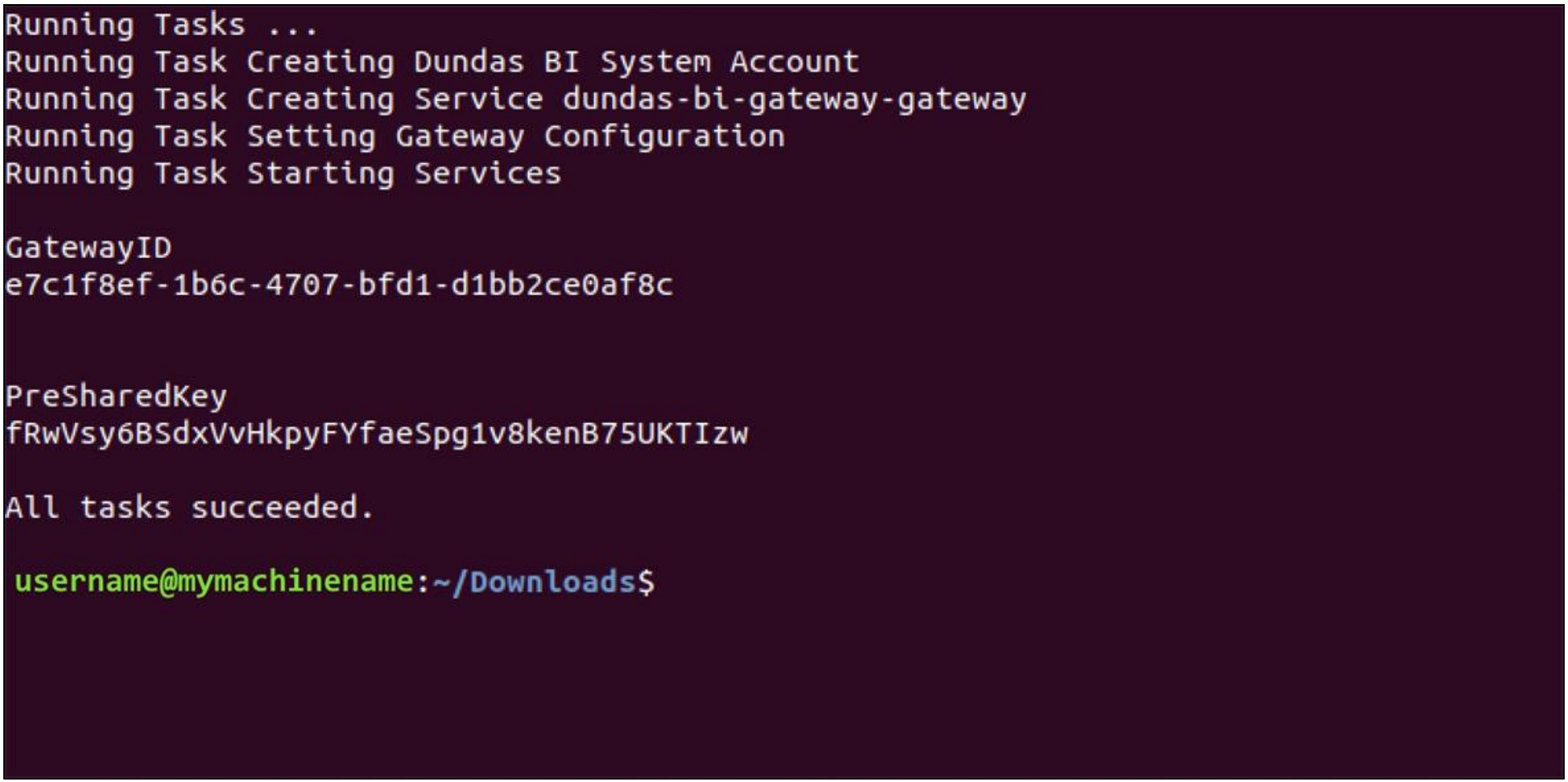


A prerequisites check may run. Follow the prompts, or type **y** and the enter key to continue if the check passed.

You will be prompted to enter a **Name** for the Symphony instance you want to use this gateway (for your reference), followed by the **SymphonyGateway Hub URL** based on the address you use to access Symphony with `/GatewayHub/` added. For example, if you can navigate to Symphony at `https://symphony.example.com/`, enter `https://symphony.example.com/managed/GatewayHub/`.



These details will be displayed back to you next and you can type **y** followed by the enter key to confirm and create the gateway.



Once the tasks are complete, note the **GatewayID** and **PreSharedKey** values indicated in your terminal, as you will need these when configuring the gateway in Symphony. If you need to find them again later, they are also defined within the **ConfigOverride.xml** file in the **App_Data** folder.

Once created, your gateway's **App_Data** directory is located at:

```
/usr/share/dundas/bi/Gateways/Files/{InstanceName}/App_Data
```

In some cases, drivers may need to be installed in subdirectories created under this directory if indicated in links from the data sources section of Symphony's system requirements. There is also a **Logs** subdirectory with files that may record errors.

A service is also installed that needs to continue running in order to provide access to your data sources for Symphony. You can control or check the status of this service using the following commands (you may need to use the sudo command in front):

Command	Result
systemctl stop dundas-bi-gateway-{InstanceName}.service	Stops the Symphony gateway service.
systemctl start dundas-bi-gateway-{InstanceName}.service	Starts the Symphony gateway service.
systemctl restart dundas-bi-gateway-{InstanceName}.service	Restarts the Symphony gateway service.
systemctl status dundas-bi-gateway-{InstanceName}.service	Get the Symphony gateway service status.

On Docker

Like other Docker images, the gateway can run as a Docker container using the image dundas/dundas-bi-gateway. This allows you to run on other machines that can access your data sources besides the supported Windows and Linux environments if compatible .

The gateway is configured using environment variables set when starting the container:

Environment Variable	Description
DUNDAS_BI_EXTERNAL_APPLICATION_URL	The URL to your instance, e.g.: https://symphony.example.com/
DUNDAS_BI_GATEWAY_ID	A unique ID to identify this gateway. You could generate a UUID for this value using uuidgen in a terminal or online.
DUNDAS_BI_GATEWAY_PRESHARED_KEY	A strong pre-shared key value matching the one you configure in your Symphony instance as described below to ensure only your gateway is granted access. Generate one or find tips here: https://cloud.google.com/network-connectivity/docs/vpn/how-to/generating-pre-shared- key
DUNDAS_BI_AGREE_TO_TERMS_OF_GATEWAY_EULA	You must accept the terms of the Gateway Application License Agreement to run the gateway. Set to **true** to confirm your acceptance.
DUNDAS_BI_PRINT_CONFIG_OVERRIDE	(Optional) Prints configuration XML text containing the gateway ID and pre-shared key you specified, as confirmation.
DUNDAS_BI_GATEWAY_JDBC_DRIVER_PATH	(Optional) The path to JDBC driver files you intend to use with the JDBC data provider, mounted as a volume or added by extending the image.

Set up the Gateway Manifest in Symphony as described in the following section first using the gateway ID and pre-shared key you plan to set as environment variables, and then start up your gateway container.

Here is an example for starting the container using docker run:

```
docker run -it \
-e DUNDAS_BI_EXTERNAL_APPLICATION_URL="https://dundas.example.com/" \
-e DUNDAS_BI_GATEWAY_ID="b3f999b0-5dc5-4790-bc9c-3a2e97d8701f" \
-e DUNDAS_BI_GATEWAY_PRESHARED_KEY="h9LR6yUxFpdXAserzdXjVxB4DQdNl3rTdKDj2Mxr" \
-e DUNDAS_BI_AGREE_TO_TERMS_OF_GATEWAY_EULA="true" \
dundas/dundas-bi-gateway
```

Set Up Symphony

Symphony must be set up for gateways before you can use them, by installing a gateway hub and configuring it for your on-premise gateway.

Install the Gateway Hub

Add the gateway hub to your Symphony instance on the server where Symphony was installed, which will add support for connecting with your gateway running on-premises.

Use the same installation tool you used to install your Symphony instance:

- When Symphony is installed directly on Linux, run the Deployment Wizard and choose the option to add the gateway hub. If you are using your own reverse proxy configuration rather than the Deployment wizard's provided one, it will need to be updated for the gateway hub.

- For Kubernetes deployments, use appropriate Helm chart options. For example:

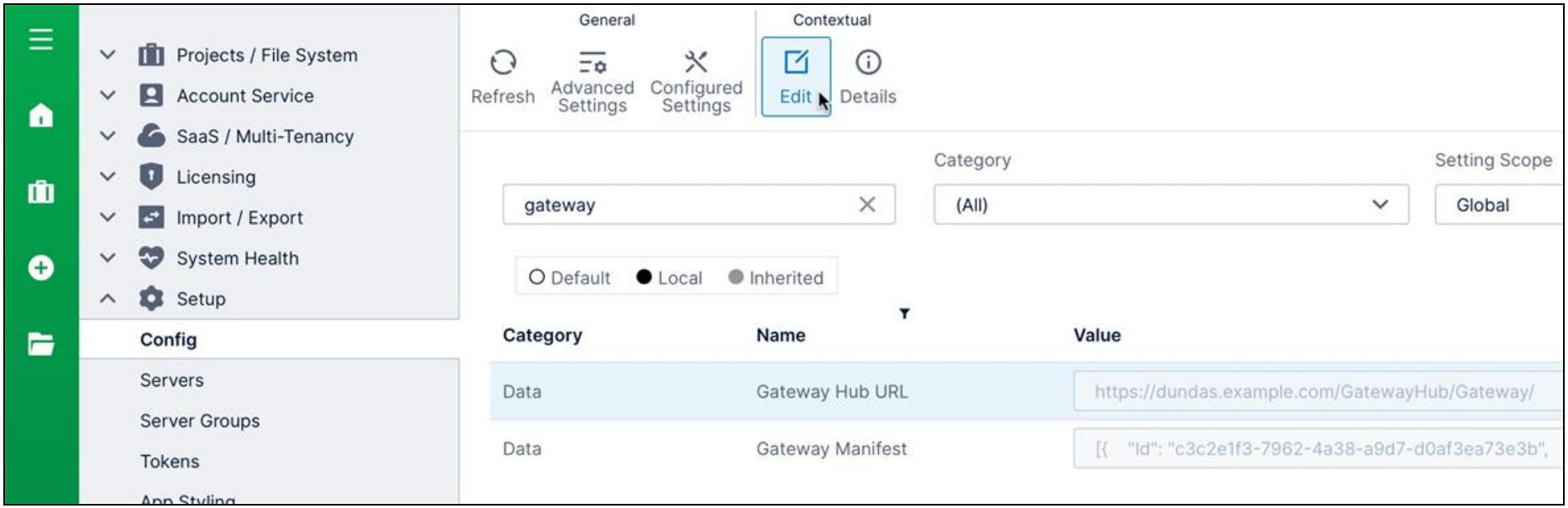
```
mananged:
  dundas:
    bi:
      gatewayhub:
        # enables the gatewayhub deployment.
        enabled: true

        # The gatewayhub service.
        service:
          enabled: true
```

- If you are running Symphony containers yourself with Docker directly, configure your reverse proxy for the dundas-bi-gateway-hub container to be accessible as the subpath `/GatewayHub` under your instance.

Configure Your Gateway

Log into Symphony as an administrator user and access the Administration area from the main menu.

Click to expand **Setup** and then to navigate to **Config** to access Symphony's configuration settings.



Find or search for the Gateway Manifest setting, located in the Data category, then double-click it or select it and click Edit.

In the configuration dialog, click Edit value, and then enter your configuration into the text area below as a JSON array like the following:

```
[{
    "Id": "c3c2e1f3-7962-4a38-a9d7-d0af3ea73e3b",
    "DisplayName": "Office",
    "PreSharedKey": "Ut5x8VGGPolxpylVmF5h5iVgmgDlphkJXiRUiTnf",
    "ipAllowList": "192.0.2.0/24"
}]
```

If you have multiple gateways, you can provide multiple comma-separated objects as defined within curly braces { }. Each object should provide the following details for a gateway:

- **Id** - The gateway ID provided to you earlier, from your gateway's Config Override File or from your terminal after adding the gateway on Linux.
- **DisplayName** - The name used to identify this gateway when using it in a new data connector.
- **PreSharedKey** - The pre-shared key value provided to you earlier along with the gateway ID.
- **ipAllowList** - (optional) If specified, incoming gateway connections to Symphony are only accepted from the specified comma-separated list of IP addresses and/or IP address ranges. An IP address range may be specified using CIDR notation (e.g., 10.10.10.0/24) or as two addresses separated by a hyphen (e.g., 10.10.10.0-10.10.10.255).

Note: `ipAllowList` is optional and may be left out, but can improve security if specified.

Click to **Save** in the dialog for the configuration to take effect.

Depending on your scenario, you may also wish to review the other Gateway configuration settings available, such as the min/max pool size and timeout settings. These can be adjusted as necessary in case of issues connecting to data through the gateway.

Configure the Gateway Hub

In configuration settings, ensure that the Gateway Hub URL is set to the `/GatewayHub/Gateway/` subpath under your Symphony instance. For example, if the address for Symphony's administration pages including configuration settings begin with `https://symphony.example.com/managed/Admin/`, the gateway hub URL should be `https://symphony.example.com/managed/GatewayHub/Gateway/`.

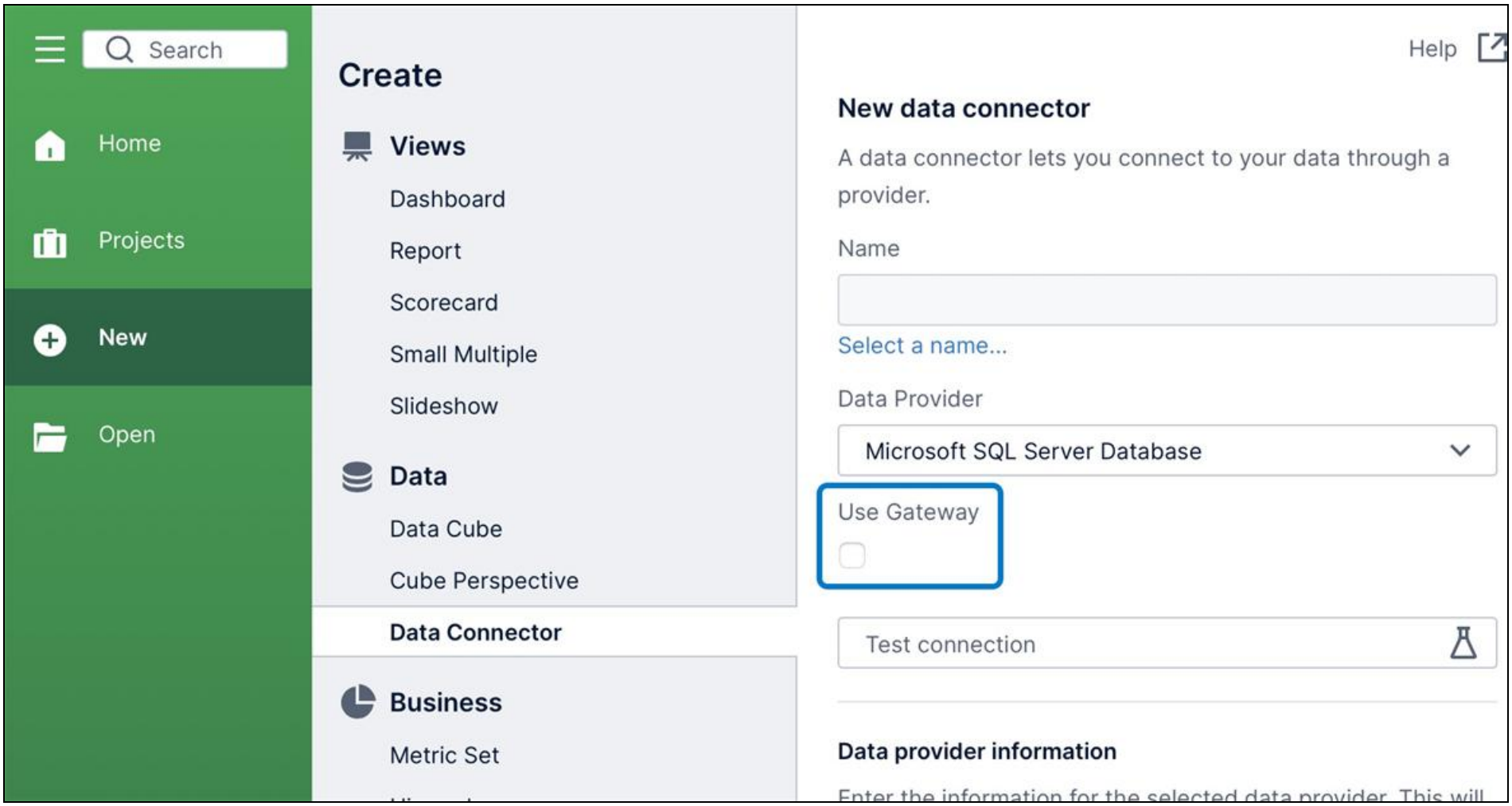
Restart Services

You may need to restart the website's application pool, or container for the changes to take effect once the gateway and gateway hub are configured.

Connect to Your Gateway

Once your gateway and gateway hub are set up and any necessary drivers are installed, users can use that gateway to connect to on- premise data sources in data connectors.

When creating or editing a data connector with Data Provider set to a supported data source, select the **Use Gateway** checkbox and then select your **Gateway** from the list that appears.



Now you can enter the connection details your gateway should use to access your data source.

Security

- The gateway uses only outgoing connections, so your environment where the gateway is installed does not need to support incoming or inbound connections for the gateway.

- Your gateway connects to the Symphony URL you specified when adding it. Your Symphony instance should be configured to use HTTPS encryption to secure this traffic with a URL that starts with https://.
- Trust is established between your Symphony instance and gateway through the pre-shared key used during configuration, so that your instance will only accept incoming connections from your gateway.
- When data source connection details need to be communicated to your gateway, a pre-shared key is used with AES encryption and a SHA-512 hash function.
- You can further improve the security for incoming gateway connections to Symphonyby specifying ipAllowList in your gateway manifest.

For more information, see:

- Connect to Data and View it on a Dashboard
- Configuration Settings
- Configuration Best Practices

Set Up and Use the Data Gateway Service

This applies to: *Visual Data Discovery*

Adding a data connector gateway to your Symphony environment allows you to connect securely to data outside of your environment. Use a gateway client to authenticate your connection and make the data available to users.

To establish communication between your data and Symphony, enable the data gateway service, SSL environment, then generate gateway clientes to retrieve your data from your external data bases, either on-premise or in the cloud.

After you've set up the data connector, users with appropriate privileges can access and use the data in source, visuals, and dashboards. Users can additionally access the data to create items in the Managed Dashboard and Reports modules.

Enable the Data Gateway Service

First, enable the data gateway service in your environment. You must be a Global user and member of the Admin group.

Important: If you have not enabled SSL, do so before completing these steps. See Enable Secure Sockets Layer.

- Symphony Installed via Kubernetes:
  - Enable `dataGatewayService` (set to `true`) in the Data Discovery section of the values.yaml. See Enable the Data Gateways and Install the Gateway Hub.
  - Two services are added and exposed through the default ingress once you enable them:
    - `data-gateway-service-external` - looks externally to accept websocket connections
    - `data-gateway-service-internal` - looks internally to represent connecters inside the cluster
  - Start the service.

Once enabled, use the API to create data gateway clients, one for each connection you want to establish, using the `/api/data-gateway/clients` endpoint. Provide a name and description as needed. ComposerSymphony returns the client `id` and client `secret`, used to authenticate to ComposerSymphony.

API documentation is provided with your Symphony installation at this link: *`<symphony-URL>/discovery/swagger-ui.html`*.

Use the Data Gateway Service

After enabling the data gateway service and creating one or more gateway clients, use credentials and the client id and client secret for each database to establish the connections from the database to ComposerSymphony.

Manage Connectors

At some point, the connector shows up on the manage connector servers page. After that you can make a connection.

Important: Once established, data gateway connectors are available in all tenants to users with appropriate privileges who can create connectors. Limit access in a multi-tenant environment as needed.

See Establish a Data Gateway Connection.

Enable Secure Sockets Layer

If you have not enabled SSL, enable it now.

Set SSL to `true` through your preferred software, such as Spring Boot. Pass the key store and the key store password. Once complete, you can connect to the gateway service through a secured Websocket URL connection. Specify this setting as a parameter in the data gateway agent.

Establish a Data Gateway Connection

This applies to: *Visual Data Discovery*

Once you or your system administrator has enabled the data gateway and added a connector, you can make a connection your users can select when creating a new data source.

Users will see the new connector as an option during source creation, and can simply select it as they would any other source. See Define a SourceDefine a Visual Data Discovery Source.

Note: You are not prompted for credentials when you create a connection: they were provided as part of defining the connector.