

Sitecore XP 10.2.0 Developer Workstation Deployment With Docker

How to install a Sitecore Experience Platform 10.2.0 on a developer workstation using containers with Docker

November 3, 2021



Table of Contents

- 1. Introduction to Docker Compose 3
 - 1.1. Supported Sitecore topologies for Docker 3
 - 1.1.1. XP Workstation (XP Single) 3
 - 1.1.2. XM Server (XM Scaled) 4
 - 1.1.3. XP Server (XP Scaled) 4
 - 1.2. Sitecore Docker Compose requirements 5
 - 1.2.1. Software requirements 5
 - 1.2.2. Hardware requirements 6
 - 1.2.3. Network requirements 6
 - 1.3. Prepare for deploying 7
 - 1.3.1. Understanding environment variables 7
 - 1.3.2. Using non-production container images 7
 - 1.3.3. Run script to prepare for deployment 8
- 2. Deploy a workstation 11
 - 2.1. Rebuild the search indexes 12
 - 2.2. Deploy custom modules 12
 - 2.2.1. Add database updates to a module 12
 - 2.2.2. Add Solr collections to a module 12
 - 2.3. Removing the Docker environment 13
- 3. Appendices 14
 - 3.1. Environment variable list 14
 - 3.2. Common issues 15
 - 3.2.1. I cannot upload a Translations file to the website root folder 15
 - 3.2.2. I can only see the main Sitecore log files for the Sitecore roles containers 16

1. Introduction to Docker Compose

Sitecore Experience Platform uses Docker Compose as the container orchestrator on developer workstations. Docker Compose is a simple container deployment tool that is bundled with Docker for Windows. You can use other tools to deploy Sitecore container images but we recommend that you use Docker Compose to deploy the containers that form the Sitecore Experience Platform.

This chapter contains the following sections:

- [Supported Sitecore topologies for Docker](#)
- [Sitecore Docker Compose requirements](#)
- [Prepare for deploying](#)

1.1. Supported Sitecore topologies for Docker

You can install Sitecore XP on developer workstations using Docker containers.

Sitecore XP for Docker supports the following topologies:

- XP Workstation (XP Single)
- XM Server (XM Scaled)
- XP Server (XP Scaled)

All three topologies are included in the Sitecore Container Deployment package you can download from the [Sitecore download page](#).

1.1.1. XP Workstation (XP Single)

The Sitecore Experience Platform Workstation for Docker topology, also known as *XP0*, is for developer workstation environments only. This topology is designed to reduce memory overhead, reduce download size, improve startup/shutdown time, and reduce complexity.

The XP0 topology supports the following Sitecore roles:

Role type	Sitecore role
Production	Sitecore Identity Server
Non-production	Content Management (Standalone)
	xConnect Server (Standalone)
	xConnect Search Indexer
	xDB Automation Engine
	Cortex Processing Engine
	Microsoft SQL Server
	Apache Solr

Role type	Sitecore role
	RedisLabs Redis Server
	Traefik Reverse Proxy

For a list of the supported environment variables, in the Container Deployment Package, in the Docker Compose folder for the XP0 topology, see the environment variable configuration file.

1.1.2. XM Server (XM Scaled)

The Sitecore Experience Manager Server for Docker topology, also known as *XM1*, is suitable for use in both production and non-production environments.

NOTE

To reduce deployment time and lower the resource overhead in non-production environments, you can remove the Content Delivery role from the Docker Compose configuration.

The XM1 topology supports the following Sitecore roles:

Role type	Sitecore role
Production	Content Management
	Content Delivery
	Sitecore Identity Server
Non-production	Microsoft SQL Server
	Apache Solr
	RedisLabs Redis Server
	Traefik Reverse Proxy

For a list of the supported environment variables, in the Container Deployment Package, in the Docker Compose folder for the XM1 topology, see the environment variable configuration file.

1.1.3. XP Server (XP Scaled)

The Sitecore Experience Platform Server for Docker topology, also known as *XP1*, is suitable for use in both production and non-production environments.

The resources required to run the XP1 topology in a non-production environment can be significant but are required to mimic the exact configuration that is used in production environments. In non-production environments, it is best practice to use a workstation that meets the minimum [workstation hardware requirements](#).

The XP1 topology supports the following Sitecore roles:

Role type	Sitecore role
Production	Content Management
	Content Delivery
	Sitecore Identity Server
	xDB Processing

Role type	Sitecore role
	xConnect Collection
	xConnect Search
	xDB Automation Operations
	xDB Automation Reporting
	xDB Reference Data
	Cortex Processing
	Cortex Reporting
	xConnect Search Indexer
	xDB Automation Engine
	Cortex Processing Engine
Non-production	Microsoft SQL Server
	Apache Solr
	RedisLabs Redis Server
	Traefik Reverse Proxy

For a list of the supported environment variables, in the Container Deployment Package, in the Docker Compose folder for the XP1 topology, see the environment variable configuration file.

1.2. Sitecore Docker Compose requirements

There are a number of requirements that your environment must fulfill before you can deploy containers with Sitecore Docker compose. These include:

- [Software requirements](#)
- [Hardware requirements](#)
- [Network requirements](#)

1.2.1. Software requirements

You must have the following software installed in order to install Sitecore Experience Platform on Docker:

- One of the following operating systems:
 - Windows 10 1903 or later. If you need to enable process isolation, you must have Windows 10 1909 or later.
 - Windows Server 1903 or later.

NOTE

For more information about Windows and containers, see [Microsoft's documentation](#).

- [Docker Desktop for Windows](#)

In addition, you must download the Sitecore Container Deployment Package from the [Sitecore download page](#).

The package includes, among other things, two configuration files that are required by Sitecore Docker Compose:

- `docker-compose.yml`
A Docker Compose configuration file that contains information about the different containers and configuration of each Sitecore role.
Studying this file can help you understand how the containers and the connection strings between the different roles function.
- `.env`
An environment variable configuration file that contains the configuration information for the environment you want to deploy. You can edit this file outside the main Docker Compose configuration.

1.2.2. Hardware requirements

The recommended minimum requirements for your workstation in order for it to run Sitecore Experience Platform on Docker are:

- RAM
For the *XP1* server topology, we recommend a developer workstation with 32GB of RAM.
For the *XM1* and *XP0* server topologies, we recommend a developer workstation with a minimum of 16GB of RAM.
- CPU
We recommend a quad-core processor or higher.
- Disk
The Sitecore container images require approximately 25 GB free space. It is best practice to use SSD disks for optimal performance when downloading and running Docker containers. The type of disks used for SQL Server and Solr can also have a significant impact on performance.

1.2.3. Network requirements

Before you deploy the Sitecore containers you must ensure that the following required TCP ports are available:

Required port	Role	Description
443	Traefik	HTTPS proxy
8079	Traefik	Traefik dashboard
8984	Solr	Solr API and dashboard
14330	SQL	SQL Server

1.3. Prepare for deploying

Before you start the process of deploying the Sitecore XP containers, there are some concepts and procedures you need to be familiar with.

These concepts and procedures are:

- [Understanding environment variables](#)
- [Using non-production container images.](#)
- [Run script to prepare for deployment](#)

1.3.1. Understanding environment variables

Environment variables are the preferred mechanism for passing configuration settings into Sitecore containers.

The environment variable configuration file `.env` contains all the environment variables. Docker Compose loads these automatically during startup. The `.env` file for the Sitecore deployment is included in the Sitecore Container Deployment package.

IMPORTANT

Each environment variable must fit inside a 30,000 character block in the `.env` file. If the size of the variable exceeds 30,000 characters, the system will **not deploy successfully**.

If you want to reuse environment variables across multiple environments, you must set the environment variables in the Windows OS and remove the corresponding keys from the environment variable configuration file that is used by Docker Compose.

1.3.2. Using non-production container images

To help developers get started quickly, Sitecore uses container images for the required services.

NOTE

The `.env` file in the Sitecore Experience Platform container package contains the information you need to access the images for your chosen version and topology.

The images include:

- Sitecore roles
- Third party software - SQL Server, Redis, and Solr
- Proxy service - Traefik

WARNING

These images are for *non-production* use only.

The non-production images are not supported by Sitecore in a production environment. The non-production services do not follow the recommended best practices for hosting a production environment and should not be considered as a basis for production environments.

Sitecore uses non-production Docker images for Microsoft SQL Server, Apache Solr, traefik from Traefik Labs, and RedisLabs Redis that are only for use on developer workstations. These images are preloaded with the required database and search configurations specific to each product and are designed to facilitate rapid deployment.

1.3.3. Run script to prepare for deployment

The Sitecore container deployment package provides a script for automating some of the preparations for Sitecore deployment to a developer machine. The `compose-init.ps1` file is located with the `.env` file.

The `compose-init.ps1` file performs the following actions:

- compresses the Sitecore license file
- creates the Identity Server token signing certificate
- populates the `.env` file
- generates TLS/HTTPS certificates
- installs the root certificate into your Trusted Root Certification Authorities
- updates Windows host names

The script accepts a number of parameters. The following table shows the parameters you can use.

Parameter	Description	Example	Mandatory
<code>LicenseXmlPath</code>	Path to the Sitecore license file. You must specify this.	<code>-LicenseXmlPath "c:\license.xml"</code>	Yes
<code>Topology</code>	The name of your chosen topology.	<code>-Topology "XM1"</code>	No
<code>IdHost</code>	A custom host name for the Identity Server.	<code>-IdHost mySCID_1</code>	No
<code>CdHost</code>	A custom host name for the CD server.	<code>-CdHost mySCCD_1</code>	No
<code>CmHost</code>	A custom host name for the CM server.	<code>-CmHost mySCCM_1</code>	No
<code>SitecoreAdminPassword</code>	A custom password for the admin user.	<code>-SitecoreAdminPassword df8732V45GH</code>	No
<code>SqlSaPassword</code>	A custom password for the database SA users.	<code>-SqlSaPassword poi3412xGH45</code>	No

Parameter	Description	Example	Mandatory
EnvFilePath	Path to the .env file. Specify this if the .env file is located in a different folder.	-EnvFilePath "c:\compose\.env"	No
SitecoreGalleryRepositoryLocation	Path to the location of the Sitecore Gallery repository. Specify this if you are not using the official repository.	-SitecoreGalleryRepositoryLocation "https:\\<path_to_repository>"	No
CertDataFolder	Path to the folder where the traefik certificates are located. Specify this if the certificates are not located in .\traefik\certs.	-CertDataFolder "c:\compose\certs"	No
SqlServer	Name of the Sql server instance (only for upgrade.env file)	-SqlServer "mssql"	No
SqlUserName	Name of the user of sql server (only for upgrade.env file)	-SqlUserName "sa"	No
IsAlwaysEncrypted	Option for sql server which allows using of encryption (only for upgrade.env file)	-IsAlwaysEncrypted \$false	No
ProcessingEngineTasksDatabaseUsername	Name of the least privileged user in database (only for upgrade.env file)	- ProcessingEngineTasksDatabaseUsername "dbo"	No

If you do not specify passwords for the `admin` or `SA` users, the script assigns default values to them.

To use the script:

1. Open a Windows command window with administrator access. Navigate to the folder containing the `compose-init.ps1` file.
2. Run the script. For example:

```
.\compose-init.ps1 -Topology "<your_topology>" -LicenseXmlPath  
"<path_to_the_license_file>" -IdHost <custom_id_host_name> -CdHost <custom_cd_host_name> -  
CmHost <custom_cm_host_name> -SitecoreAdminPassword <your_password_for_Sitecore_admin> -  
SqlSaPassword <your_password_for_sql_sa_user>
```

NOTE

If your Traefik certificates folder already contains certificates the script does not install new certificates.

2. Deploy a workstation

You use Docker for Windows to deploy the Sitecore Experience Platform (SXP) container packages.

To deploy an SXP developer workstation on containers:

1. In Docker for Windows, [switch to Windows container mode](#).
2. Download the SXP Container Deployment Package from the [Sitecore download page](#) and extract it to a folder on your local workstation. Navigate to the `compose\<windows version>\<topology>` folder for the topology that you want to deploy, for example, `compose\ltsc2019\xp1`.
3. In the topology folder, run the `compose-init.ps1` [script](#).
4. In the Windows console, go to the folder that the `docker-compose.yml` file is in and run the following Docker Compose command:

```
docker-compose.exe up --detach
```

Docker Compose pulls all the required images from the Sitecore Container Registry, creates the required Docker network configuration, and deploys all the containers to the local environment.

NOTE

The required images include Sitecore roles and third party services for your chosen topology.

When the deployment is successfully completed, the Docker Compose command exits.

5. To check the Docker container status, run the following command:

```
docker-compose.exe ps
```

This command generates a list of all the containers and their current status.

6. When the status of all the containers is listed as *healthy*, open a browser and enter the URL for the content management instance.

The default URLs for, for example, the XP1 topology are:

- `https://xp1cm.localhost`
- `https://xp1cd.localhost`
- `https://xp1id.localhost`

The content management, content delivery, and identity services instances use the HTTPS protocol on port 443. If this port is in use by another process, you get an error message.

7. Verify that there are no errors in the internal container log files.

2.1. Rebuild the search indexes

When the deployment is done, you must rebuild the search indexes.

To rebuild the search indexes:

1. On the Sitecore Launchpad, open the Control Panel.
2. In the **Indexing** section, click **Populate Solr Managed Schema**.
3. In the **Schema Populate** dialog box, click **Select All**, then click **Populate**. Wait for the process to finish.
4. On the Control Panel, in the **Indexing** section, click **Indexing Manager**.
5. In the **Indexing Manager** dialog box, click **Select All**, then click **Rebuild**.

2.2. Deploy custom modules

For Sitecore components, by default, you deploy only the Solr collections and dacpacs included in the platform. In Sitecore 10.1 and later versions you can also deploy custom modules.

If the module you want to deploy requires database updates and/or custom Solr collections, you might need to build a new layer on top of the `mssql-init` and/or `solr-init` images.

2.2.1. Add database updates to a module

To add database updates to a module:

1. Build a new layer on top of the `mssql-init` image.
2. In your build-script, add a command for the following action:
 - Copy the module dacpacs from the module asset image into the `c:\module_name_data` folder for the new image.

2.2.2. Add Solr collections to a module

To add Solr collections to a module:

1. Build a new layer on top of the `solr-init` image.
2. Add the Solr collections module configuration files from the module assets image to the `c:\data` folder.

NOTE

You must use the JSON file format for collections. If the module name is `sxa`, you can, for example, name the Solr collections file `cores-sxa.json`.

2.3. Removing the Docker environment

With [Docker Compose commands](#), you can stop, resume, or remove a workstation environment.

The basic commands are:

- To stop a Docker Compose environment without removing its contents:

```
docker-compose.exe stop
```

- To resume a previously stopped Docker Compose environment:

```
docker-compose.exe start
```

- To remove a Docker Compose environment and all the non-mounted volumes:

```
docker-compose.exe down
```

3. Appendices

This section contains additional information and helper functions to help you with deployment to Docker.

The topics include:

- [Environment variable list](#)
- [Common issues](#)

3.1. Environment variable list

The following table describes the environment variables and lists their default values.

Variable name	Default value	Description
SITECORE_DOCKER_REGISTRY	scr.sitecore.com/ sxp/	Sitecore container registry
SITECORE_VERSION	10.2.0-1tsc2019	Image tag with the version to be pulled from the container registry
SITECORE_ADMIN_PASSWORD		Sitecore application administrator password
SQL_SA_PASSWORD		SQL Server administrator password
REPORTING_API_KEY		Symmetric key used to access the Sitecore XDB Processing Service. Length: 64-128 characters
TELERIK_ENCRYPTION_KEY		Symmetric key used by the Telerik web controls. Length: 64-128 characters
SITECORE_IDSECRET		Shared secret between the Identity Server and client roles. Length: 64 characters
SITECORE_ID_CERTIFICATE		Identity Server certificate used to encrypt data
SITECORE_ID_CERTIFICATE_PASSWORD		Password to open the Identity Server certificate
SITECORE_LICENSE		License file content converted to GZIP Compressed and Base64 encoded string

Variable name	Default value	Description
ISOLATION	default	Override for Docker isolation level (Possible values: default, hyperv, process)
SOLR_CORE_PREFIX_NAME	sitecore	A common prefix for Solr core names. If you use an existing Solr deployment, you must change this to your actual Solr core name prefix.
MEDIA_REQUEST_PROTECTION_SHARED_SECRET		Shared secret. You must change this to a random string. Do not use the default value.

3.2. Common issues

This section lists some common issues related to the installation of a Developer Workstation with containers, along with proposed solutions.

3.2.1. I cannot upload a Translations file to the website root folder

If the **Upload Files** dialog hangs during the upload operation, it writes the following errors to the log file:

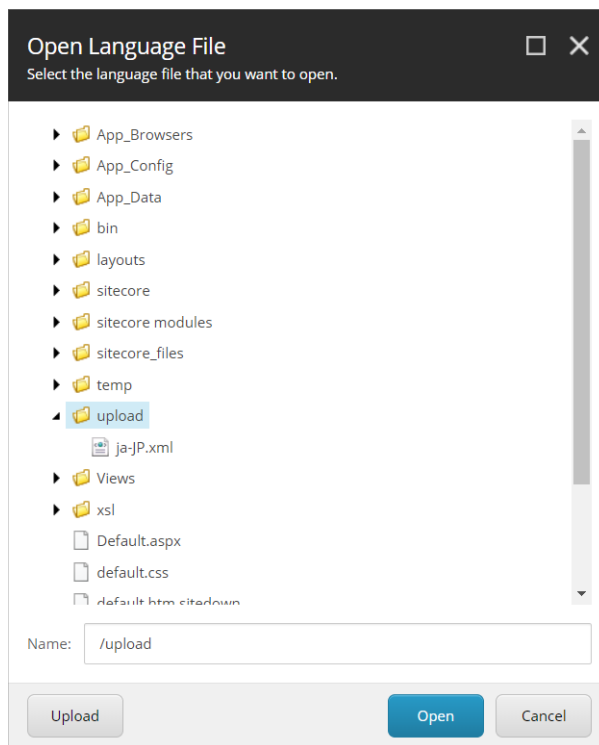
```
ERROR Could not save posted file: ja-JP.xml
Exception: System.UnauthorizedAccessException
Message: Access to the path 'C:\inetpub\wwwroot\ja-JP.xml' is denied.
```

This happens because the **Import language** dialog box uploads translations files to the `website` root folder by default and for security reasons *Write* access is denied for the website root folder.

The `upload` folder, however, has *Write* access enabled. To resolve this issue, upload the translations files to the `upload` folder.

To upload a translations file:

1. In the **Open Language File** dialog box, select the `upload` folder and then click **Upload**.



2. In the **Upload Files** dialog box, upload the translations file to the `upload` folder.
3. Import the translations file from the `upload` folder.

NOTE

The translations xml file is saved to the Media library. You can delete it after you import the translations.

3.2.2. I can only see the main Sitecore log files for the Sitecore roles containers

The LogMonitor tool collects the log files for containers. By default, it monitors the following log files:

- System event log – error level entries
- IIS logs
- Primary Sitecore log – `log.*.txt` files for the Sitecore roles
- xConnect log – `xconnect-log-*.txt` files for the xConnect roles

Auxiliary Sitecore log files, such as for search, crawling, and publishing, are not monitored on Sitecore containers.

To see all the Sitecore log files for a Sitecore role container, you must create a Dockerfile with the corresponding role image and reconfigure the LogMonitor tool or replace the entire configuration file with the updated configuration.

To reconfigure the LogMonitor tool:

1. In the `c:\inetpub\wwwroot\App_data\logs` folder, open the `C:\LogMonitor\LogMonitorConfig.json` file.

2. In the sources node, edit the filter setting:

```
{
  "LogConfig": {
    "sources": [
      ...
      {
        "type": "File",
        "directory": "c:\\inetpub\\wwwroot\\App_data\\logs",
        "filter": ".*log*.txt",
        "includeSubdirectories": false
      }
    ]
  }
}
```

Now you can use the Dockerfile to build a new docker image for the Sitecore role.

You can also view all the Sitecore log files directly from the container's file system by connecting to the corresponding container from a PowerShell or command prompt terminal.