Installation Guide for a Developer Workstation with Containers

Sitecore XP 10.1.0

A quick guide to installing a developer workstation with containers

Contents

Chapter 1

Introduction

Sitecore Experience Platform 10.1.0 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. Sitecore container images can be deployed with other tools but we recommend that you use Docker Compose to deploy the containers that form the Sitecore Experience Platform.

This chapter contains the following sections:

- Topologies
- Sitecore Docker Compose
- Preparation

1.1 Topologies

You can install Sitecore XP 10.1.0 on developer workstations using containers.

Sitecore XP 10.1.0 supports the following topologies:

XP Workstation (XP Single)

The Sitecore Experience Platform workstation for Docker 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 XP Workstation (XP Single) topology supports the following Sitecore roles:

Role Type	Sitecore Role
Production roles	Sitecore Identity Server
Non-production roles	Content Management (Standalone)
	xConnect Server (Standalone)
	xConnect Search Indexer
	xDB Automation Engine
	Cortex Processing Engine
	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 XP Single topology, see the environment variable configuration file.

XM Server (XM Scaled)

The Sitecore Experience Manager Server for Docker is suitable for use in production and non-production environments. 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 XM Server (XM Scaled) topology supports the following Sitecore roles:

Role Type	Sitecore Role
Production roles	Content Management
	Content Delivery
	Sitecore Identity Server
Non-production roles	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 XM Scaled topology, see the environment variable configuration file.

XP Server (XP Scaled)

Sitecore Experience Platform Server for Docker is suitable for production and non-production environments.

The resources required to run the XP Server 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, we recommend that you run XP Server with the workstation hardware requirements.

The XP Server (XP Scaled) topology supports the following Sitecore roles:

Role Type	Sitecore Role
Production roles	Content Management
	Content Delivery
	Sitecore Identity Server
	xDB Processing
	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 roles	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 XP Scaled topology, see the environment variable configuration file.

1.2 Sitecore Docker Compose

To deploy the containers, Sitecore Docker Compose requires two text files:

- `docker-compose.yaml`

  A Docker Compose configuration file that contains information about the different containers and configuration of each Sitecore role.

- `.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.

The Docker Compose files for each Sitecore topology are included in the *Sitecore Container Deployment Package* that you can download from https://dev.sitecore.com

1.2.1 Software Requirements

- Operating System
  - Windows 10 1809 or later

    or

  - Windows Server 1809 or later
- Docker Desktop for Windows
- Docker Desktop Enterprise can be used on Windows Server
- Sitecore Container Deployment Package

  You must extract the Docker Compose configuration folder for the desired topology.

1.2.2 Hardware Requirements

- RAM

  We recommend that a developer workstation has 32GB of RAM. That is enough to run the Server (XP1) topology.

  A minimum of 16GB of RAM is required to run the XM Server (XM1) topology.

- CPU

  We recommend a quad core or higher.

- Disk

  The Sitecore container images require approximately 25 GB free space. We recommend SSD disks for optimal performance when downloading and running Docker containers. The disks used for SQL Server and Solr can have a significant impact on performance.

1.2.3 Networking Requirements

Before you can deploy the Sitecore containers you must ensure that the 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 Preparation

Before you deploy the Sitecore XP containers, you must have some requirements in place.

You must prepare the following requirements:

- Environment variables.
- Sitecore license file.
- Sitecore Identity Server token signing certificate.
- TLS/HTTPS certificates.
- Windows hosts file.

1.3.1 The 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 and Docker Compose loads these automatically during startup.

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

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

1.3.2 The Sitecore license file

Sitecore license files are typically passed to container instances as an environment variable in string form. However, the Sitecore license file is very large and you must therefore compress and *Base64* encode it to conform with the maximum size allowed by Windows for all the environment variables.

When you have compressed and encoded the license file, copy the string value to the environment variable configuration file or set it as a Windows system environment variable.

Appendix A contains a sample PowerShell script that converts a license file into a Base64 compressed string for use in an environment variable.

Note
Some Sitecore license files are so large that they are incompatible with containers even after compression. This usually happens when additional HTML is embedded in the license file. To obtain a license file for use with containers, contact your Sitecore partner.

As a workaround, you can mount the license file as a Docker volume from the host to the `c:\inetpub\wwwroot\app_data\license.xml` file inside the container. For more information and a configuration example, see Sitecore Container Development documentation.

1.3.3 The Identity Server token signing certificate

Sitecore Identity Server requires a private key certificate to sign the tokens that are passed between the server and the clients. This certificate must be generated, Base64 encoded in string form, and passed to the container as an environment variable.

Appendix C contains a sample script that generates a self-signed certificate and prepares the string that is used as an environment variable. The sample script creates the text file with the certificate and copies the content of the file to the environment variable configuration file.

Like the Sitecore license file, you can mount the Sitecore Identity Server certificate on the file system instead of passing it as an environment variable. For more information and a configuration example, see Sitecore Container Development documentation.

1.3.4 TLS/HTTPS certificates

To satisfy modern browser requirements and provide a secure environment by default, you must generate certificates for TLS (Transport Layer Security) before you deploy the Sitecore containers. This ensures secure communication between the browser and the HTTPS reverse proxy container.

The default reverse proxy or edge router used by the Sitecore Experience Platform in Docker Compose is Traefik. The Traefik edge router is used as a reverse proxy to the individual XP containers and terminates the TLS connections sent by the browser. For more information, see the Traefik documentation about TLS configuration.

You can view the reverse proxy configuration in the Traefik dashboard. When you have completed the deployment, you can see the dashboards at http://localhost:8079.

All internal communication between the Traefik edge router and the individual XP containers is sent unencrypted with the HTTP protocol.

The HTTPS protocol is required between the reverse proxy and client browsers to support the secure browser cookies used by the Sitecore Content Management, Sitecore Content Delivery and the Identity Server roles.

Appendix D contains a sample script that generates the required certificates. This script is also available in the Sitecore Container Deployment Package. Once the self-signed root authority certificate and per-host TLS/SSL certificates have been generated, you must install the root authority certificate in the Trusted Root Certificate Authority store on all the clients.

1.3.5 The Windows hosts file

To access the Sitecore application from a browser, you must update the Windows hosts file to include the host names that are used by the reverse proxy container. The default host names vary depending on the topology you decide to deploy. All the host names must point to the loopback IP address 127.0.0.1.

The following table lists the default hostnames for each topology:

Topology	Hostnames	IP address
XM Server (XM1)	xm1cm.localhost	127.0.0.1
	xm1cd.localhost	127.0.0.1
	xm1id.localhost	127.0.0.1

Topology	Hostnames	IP address
XP Workstation (XP0)	xp0cm.localhost	127.0.0.1
	xp0id.localhost	127.0.0.1
XP Server (XP1)	xp1cm.localhost	127.0.0.1
	xp1cd.localhost	127.0.0.1
	xp1id.localhost	127.0.0.1

To change the default host names, you must:

- Generate the TLS certificates with the correct host names.
- Update the Traefik reverse proxy configuration labels for each role with the correct host names.

  For more information, see the documentation for Traefik Docker configuration discovery.

1.3.6 Production and nonproduction container images

To help developers get started quickly, Sitecore provides container images for the required services. These 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 best practices that are recommended for hosting a production environment and should not be considered as a basis for production environments.

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

Every container image that has the *type=nonproduction* label is not supported in production environments. No warranty or extended support is provided for images that are labelled for non-production.

1.3.7 Deploying custom modules

For Sitecore components, by default, you deploy only the Solr collections and dacpacs included in the platform. Starting from Sitecore 10.1.0 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 and/or solr-init images. You must use the module assets image as a source for the module dacpacs and Solr collections configuration files.

To apply module database updates:

1. Build a new layer on top of the mssql image.
2. Add commands for the following:
   - copy the module dacpacs from the module assests image into the `c:\module_name_data` folder
   - execute the `C:\DeployDatabases.ps1 -ResourcesDirectory C:\sxa_data` command
   - remove the `c:\module_name_data` folder.

To add module Solr collections:

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

> **Note**
> You must use the JSON file format for collections. For example, if the module name is `sxa`, you could name the Solr collections file `cores-sxa.json`.

You can now use the newly build mssql and solr-init images to run containers with custom modules installed.

Chapter 2

Deploying a Sitecore XP workstation

You use Docker for Windows to deploy the Sitecore XP container packages.

This chapter contains the following sections:

- Deploy a workstation

2.1 Deploy a workstation

To deploy a Sitecore XP developer workstation:

1. In Docker for Windows, switch to Windows container mode.
2. Download and extract the *Sitecore Container Deployment Package* from the Sitecore Developer Portal and store it on your local workstation.
3. In Windows Explorer go to the folder that you extracted the Sitecore Container Deployment Package to and open the Docker Compose folder for the topology that you want to deploy.
4. Update the environment configuration file with the appropriate values for all the environment variables including the required passwords, encryption keys, certificates, and the Sitecore license file.

   For more information, see the environment variable list in *Appendix B*.
5. To generate the required TLS reverse proxy certificates, execute the sample PowerShell script.

   This script is provided in the *Sitecore Container Deployment Package* and documented in *Appendix D.*
6. Install the self-signed root authority certificate that you just generated in the local Trusted Root Certification Authority certificate store on your local computer. The generated root certificate path is *./traefik/certs/rootca.crt*.
7. Open the `docker-compose.yaml` file.

   Studying this file gives you a better understanding of the containers and connection strings between the different roles
8. In the Windows console, go to the folder that the `docker-compose.yaml` 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.

   When the deployment is successfully completed, the Docker Compose command exits.
9. 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.
10. When the status of all the containers is listed as *healthy*, open a browser and enter the URL for the content management instance.

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

    The default host name for the XP workstation is:

    ```
    https://xp0cm.localhost
    ```

    The default host names for the other topologies are listed in the section *The Windows hosts file.*
11. Log in to the Sitecore admin section and test the configuration.

12. Open the Sitecore Control Panel and populate the managed schema for *all* the search indexes.
13. From Sitecore Control Panel rebuild the search indexes for *all* the databases.

2.1.1 Cleanup the workstation

You can stop or completely remove a workstation environment by running Docker Compose commands.

- 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
```

Chapter 3

Appendix

3.1 Appendix A

The License file compression and encoding PowerShell helper function

```
function ConvertTo-CompressedBase64String {
    [CmdletBinding()]
    Param (
        [Parameter(Mandatory)]
        [ValidateScript( {
                if (-Not ($_ | Test-Path) ) {
                    throw "The file or folder $_ does not exist"
                }
                if (-Not ($_ | Test-Path -PathType Leaf) ) {
                    throw "The Path argument must be a file. Folder paths are not
allowed."
                }
                return $true
            })]
        [string] $Path
    )
    $fileBytes = [System.IO.File]::ReadAllBytes($Path)
    [System.IO.MemoryStream] $memoryStream = New-Object System.IO.MemoryStream
    $gzipStream = New-Object System.IO.Compression.GzipStream $memoryStream,
([IO.Compression.CompressionMode]::Compress)
    $gzipStream.Write($fileBytes, 0, $fileBytes.Length)
    $gzipStream.Close()
    $memoryStream.Close()
    $compressedFileBytes = $memoryStream.ToArray()
    $encodedCompressedFileData = [Convert]::ToBase64String($compressedFileBytes)
    $gzipStream.Dispose()
    $memoryStream.Dispose()
    return $encodedCompressedFileData
}
ConvertTo-CompressedBase64String -Path .\license.xml
```

3.2 Appendix B

Environment variable list

Variable name	Default Value	Description
SITECORE_DOCKER_REGISTRY	scr.sitecore.com/sxp/	Sitecore container registry
SITECORE_VERSION	10.1.0-ltsc2019	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 (See Appendix C)
SITECORE_ID_CERTIFICATE_PASSWORD		Password to open the Identity Server certificate

Variable name	Default Value	Description
SITECORE_LICENSE		License file content converted to GZIP Compressed and Base64 encoded string (See Appendix A)
ISOLATION	default	Override for Docker isolation level (Possible values: default, hyperv, process)
SOLR_CORE_PREFIX_NAME	sitecore	A common prefix for solr cores names. If you use an existing Solr deployment, you must update this to your actual solr cores names prefix.
MEDIA_REQUEST_PROTECTION_SHARED_SECRET	HQ(NjM(u6_5koVla-cTf4ta8x1h6Sb+ZcUQrULUz-0Afpx0cx-NuMtIoQkpDFmX5	Shared secret that you should change to a random string. Do not use the default value.

3.3 Appendix C

Create the Identity Server token signing certificate

Use the following PowerShell script to create the Identity Server token signing certificate:

```
$newCert = New-SelfSignedCertificate -DnsName "localhost" -FriendlyName "Sitecore Identity
Token Signing" -NotAfter (Get-Date).AddYears(5)

Export-PfxCertificate -Cert $newCert -FilePath .\SitecoreIdentityTokenSigning.pfx -Password
(ConvertTo-SecureString -String "Test123!" -Force -AsPlainText)

[System.Convert]::ToBase64String([System.IO.File]::ReadAllBytes((Get-Item
.\SitecoreIdentityTokenSigning.pfx))) | Out-File -Encoding ascii -NoNewline -Confirm -FilePath
.\SitecoreIdentityTokenSigning.txt
```

3.4 Appendix D

How to generate TLS/SSL certificates used by reverse proxy

Open a Windows Command Prompt as Administrator in the same folder as the docker-compose.yml file. To generate the TLS/SSL certificates that are required by the Traefik reverse proxy container, execute the following commands:

```
IF NOT EXIST mkcert.exe powershell Invoke-WebRequest
    https://github.com/FiloSottile/mkcert/releases/download/v1.4.1/mkcert-v1.4.1-windows-
    amd64.exe -UseBasicParsing -OutFile mkcert.exe
mkcert -install
del /Q /S traefik\certs\*
mkcert -cert-file traefik\certs\xm1cm.localhost.crt -key-file
    traefik\certs\xm1cm.localhost.key "xm1cm.localhost"
mkcert -cert-file traefik\certs\xm1id.localhost.crt -key-file
    traefik\certs\xm1id.localhost.key "xm1id.localhost"
mkcert -cert-file traefik\certs\xm1cd.localhost.crt -key-file
    traefik\certs\xm1cd.localhost.key "xm1cd.localhost"
```

"The mkcert utility will prompt the user the first time to install the generated self-signed root certificate authority.

3.5 Appendix E – Common issues

I cannot upload a Translations file to the website root folder

If the **Upload Files** dialog box hangs during the upload operation, the following errors are written to the log file:

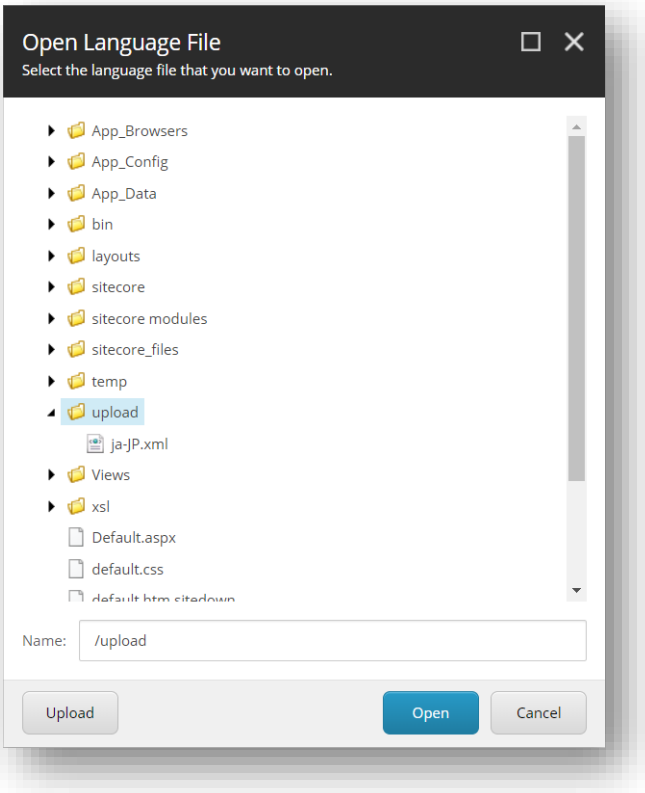
```
ERROR Could not save posted file: ja-JP.xml
Exception: System.UnauthorizedAccessException
Message: Access to the path 'C:\inetpub\wwwroot\ja-JP.xml' is denied.
```

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.

To resolve this issue, you should upload the translations files to the `\upload\` folder. *Write* access is enabled for this 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, you can now upload the translations file to the `\upload\` folder.

You can now import the translations file from this folder.

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

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

The LogMonitor tool is used to collect the log files for containers.

The LogMonitor tool is configured to monitor 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 search, crawling, publishing, and so on 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.