SITECORE

Sitecore Publishing Service Container Deployment Guide

How to deploy SPS in a container with Docker Compose or with Kubernetes

Sitecore Experience Platform 10.4.0

Table of Contents

1. Sitecore Publishing Service deployment

Introduction to deploying Sitecore Publishing Service to a container environment

To deploy the Sitecore Publishing Service (SPS) as a container in a cloud-based solution, you must deploy both the module and the service on top of your Sitecore XP 10.3.0 solution. You can use either Docker Compose or Azure Kubernetes Service.

This document describes how to use both tools to deploy the Sitecore Publishing Service container with the Sitecore Publishing Service 7.0 and the Sitecore Publishing Service Module 10.4.0. This document is only valid for Sitecore XP 10.4.0.

2. Deploy SPS with Docker Compose

Before you use Docker Compose to deploy the Sitecore Publishing Service on top of your Sitecore XP instance, you must build the custom Content Management (CM), Content Delivery (CD), and Microsoft SQL Server images that contain the Publishing Service integration data.

2.1. Prepare for the deployment

To prepare your system for the deployment:

1. Create an empty folder and name it `deploy`.
2. In the GitHub Sitecore containers deployment repository, locate and download the `SitecoreContainerDeployment.10.4.0.*.zip` package for Sitecore XP 10.4.0.
3. Unzip the package and locate the `compose/ltsc2019` folder.
4. Extract the container configuration for the topology that you are going to deploy, for example, *XP1*, to the `deploy` folder.
5. Copy the `docker-compose.yml` file to the `deploy` folder.
6. From the same repository, download the `SitecorePublishingServiceContainerDeployment.10.4.0.*.zip` package for Sitecore XP 10.4.0.
7. Extract the container configuration for the appropriate operating system to the `deploy` folder.
8. Copy the `docker-compose.sps.override.yml` file to the `deploy` folder.

2.2. Build the custom images

You must prepare the CM, CD, and MSSQL custom images before you deploy the containers and these images must include all the resources – files, databases, and so on – from the Sitecore Publishing Service asset image.

The asset images are:

- sxp/modules/sitecore-sps-integration-xm1-assets
- sxp/modules/sitecore-sps-integration-xp1-assets

You must use the asset image for the topology you are deploying.

For more information about how to apply the asset images, see the following articles:

- Add Sitecore modules
- Sitecore module reference

You must also prepare a custom Publishing Service image that includes all master and web resource files from the CM image.

> **IMPORTANT**
> This step should be done against the custom-built CM image with all asset images applied to make sure the resource files from Sitecore and Modules are all copied into the Publishing Service image. For example, if you are deploying Sitecore with SXA, JSS, and Publishing Service modules, you must apply the three modules asset images first then use the built CM to build the custom Publishing Service image.

A typical docker file should look like the following where the SPS_IMAGE argument refers to the Publishing Service image tag and SITECORE_IMAGE refers to the custom-built CM image tag:

```
# escape=`
ARG SPS_IMAGE
ARG SITECORE_IMAGE
FROM ${SITECORE_IMAGE} AS sitecore
RUN mkdir c:/res_files
WORKDIR c:/res_files
RUN Copy-Item c:/inetpub/wwwroot/App_Data/items/master sitecore/master -Recurse -Force; `
Copy-Item c:/inetpub/wwwroot/App_Data/items/web sitecore/web -Recurse -Force; `
if(Test-Path -Path \"c:/inetpub/wwwroot/sitecore modules/items/master\") `
{ `
Copy-Item \"c:/inetpub/wwwroot/sitecore modules/items/master\" modules/master -Recurse -Force; `
} `
if(Test-Path -Path \"c:/inetpub/wwwroot/sitecore modules/items/web\") `
{ `
Copy-Item \"c:/inetpub/wwwroot/sitecore modules/items/web\" modules/web -Recurse -Force; `
}
FROM ${SPS_IMAGE} AS base
COPY --from=sitecore c:/res_files/ C:/sps/items/
```

2.3. Deploy SPS on top of Sitecore XP

To deploy the SPS container on top of Sitecore XP:

1. In the deploy folder, open the .env file and populate the required Sitecore parameters. For more information about the secrets, see the Installation Guide for a Developer Workstation with Containers for the relevant Sitecore XP version.

2. Append the following parameters that are required to deploy the Publishing Service:

Variable	Description
VERSION	The version tag for the CM, CD, and MSSQL patched images.

Variable	Description
TOPOLOGY	The topology that you want to deploy. The supported values are: • xp1 • xm1

3. In the deploy folder, run the following command:

```
docker-compose -f .\docker-compose.yml -f .\docker-compose.override.sps.yml up -d
```

2.3.1. Update the local host file

You must update the local host file with the localhost IP address (127.0.0.1).

For example, the default hostnames for the XP1 topology are:

- `xp1cm.localhost`
- `xp1cd.localhost`
- `xp1id.localhost`

3. Deploy SPS with Kubernetes

In the following procedures, replace {topology} with xp1 or xm1 depending on the Sitecore topology you are deploying.

3.1. Walkthrough: preparing to deploy Publishing Service in Kubernetes

Before you deploy the SPS container with Kubernetes you must prepare your installation.

3.1.1. Download Kubernetes specification files

In the GitHub Sitecore containers deployment repository, locate and download the following files for Sitecore XP:

- `SitecorePublishingServiceContainerDeployment.10.4.0.*.zip`
  This file contains the Sitecore Publishing Service Kubernetes configuration files for Sitecore XP 10.4.0.
  Unzip the package and copy the `k8s\ltsc2019` folder into your working folder.
- `SitecoreContainerDeployment.10.4.0.*.zip`
  Unzip the archive and locate the `k8s\ltsc2019\{topology}` folder for the Sitecore topology that you want to deploy.
  Copy this folder to your working folder next to the Sitecore Publishing Service configuration files.

For example, for xp1 topology, you should have the following folder structure:

- sps
- overrides
- xp1

3.1.2. Set up Kubernetes configuration

Download the Installation Guide for Production Environments with Kubernetes and follow all the steps in the *Prerequisites* section to set up your local Sitecore Kubernetes configuration.

3.1.3. Build the images

You must prepare the Sitecore CM, CD, and MSSQL-INIT custom images and these images must include all the resources – files, databases, and so on – from the SPS asset image.

The available asset images are:

- sxp/modules/sitecore-sps-integration-xm1-assets
- sxp/modules/sitecore-sps-integration-xp1-assets

Use the asset image for the topology you are deploying.

For more information about how to apply the asset images, see:

- Add Sitecore modules
- Sitecore module reference

You must also prepare a custom Publishing Service image that includes all master and web resource files from the CM image.

> **NOTE**
> You must perform this step against the custom-built content management image with all asset images applied to make sure resource files from Sitecore and Modules are all copied into Publishing Service image. For example, if you are deploying Sitecore with SXA, JSS and Publishing Service modules, you must apply the 3 modules asset images first then use the built CM to build the custom Publishing Service image.

A typical docker file should look like the following where the `SPS_IMAGE` argument refers to the Publishing Service image tag and `SITECORE_IMAGE` refers to the custom-built content management image tag:

```
# escape=`
ARG SPS_IMAGE
ARG SITECORE_IMAGE

FROM ${SITECORE_IMAGE} AS sitecore

RUN mkdir c:/res_files
WORKDIR c:/res_files

RUN Copy-Item c:/inetpub/wwwroot/App_Data/items/master sitecore/master -Recurse -Force; `
    Copy-Item c:/inetpub/wwwroot/App_Data/items/web sitecore/web -Recurse -Force; `
    if(Test-Path -Path \"c:/inetpub/wwwroot/sitecore modules/items/master\") `
        { `
Copy-Item \"c:/inetpub/wwwroot/sitecore modules/items/master\" modules/master -Recurse -Force; `
    } `
        if(Test-Path -Path \"c:/inetpub/wwwroot/sitecore modules/items/web\") `
    { `
Copy-Item \"c:/inetpub/wwwroot/sitecore modules/items/web\" modules/web -Recurse -Force; `
    }
FROM ${SPS_IMAGE} AS base

COPY --from=sitecore c:/res_files/ C:/sps/items/
```

3.1.4. Push the images

You must push the CM, CD, SPS, and MSSQL-INIT images to a private container registry so they can be picked up later in the deployment process.

To push the images, use the Docker `push` command.

For more information about the Docker Compose commands, see the Docker CLI documentation.

3.1.5. Configure the images

To update the Sitecore Publishing Service Kubernetes specification with the Sitecore CM, CD, SPS, and MSSQL-INIT images:

1. Open the overrides\{topology}\kustomization.yaml file and change the value of the newname and newTag properties to the corresponding image name and tag for the CM and CD images.

2. Open the sps\kustomization.yaml file and change the value of the newname and newTag properties to the corresponding image name and tag for the Publishing Service image.
3. Open the overrides\{topology}\init\kustomization.yaml file and change the newname and newTag properties to the corresponding image name and tag for the MSSQL-INIT image.

3.2. Deploy the Sitecore Platform with SPS to the Azure Kubernetes server

To deploy the Sitecore XP and Sitecore Publishing Service to a Kubernetes cluster, you must use the Kubectl CLI. This chapter describes how to deploy Sitecore Experience Platform with Sitecore Publishing Service to the Azure Kubernetes Service.

3.2.1. Create an AKS cluster

To create a new Azure Kubernetes Service (AKS) cluster with a Windows Server 2019 node pool, you can use the Azure command-line interface (Azure CLI) or the Azure portal UI. The AKS cluster must contain one Windows Server 2019 version 1809 node pool with one or more nodes.

For more information about using the Azure CLI to create an AKS cluster, see the Azure AKS documentation.

3.2.2. Configure the Kubectl context cluster

1. Log in to the Azure CLI and set a subscription. For example:

```
az login az account set --subscription "Your Subscription"
```

2. Get the credentials for the K8s cluster that were created with the AKS cluster. For example:

```
az aks get-credentials --resource-group sc10aks --name sc10cluster
```

3.2.3. Configure access to a private container registry

The deployed Kubernetes cluster requires access to the private container registry where the patched CM, CD, and MSSQL-INIT images were pushed. To let a Kubernetes cluster authenticate with a private container registry and pull images from it, you must create an image pull secret. The name of the secret must be *sitecore-docker-registry*.

3.2.4. Deploy an ingress controller

To deploy an NGINX ingress controller:

1. Use the Windows AMD64 binaries to Install Helm. Helm is the package manager for Kubernetes.
   Alternatively, you can use another operating system package manager to install Helm.
2. To add an NGINX ingress controller feed to Helm, run this command:

```
helm repo add stable https://charts.helm.sh/stable
```

3. To update the feed, run this command:

```
helm repo update
```

4. To use Helm to deploy the NGINX ingress controller, run this command:

```
helm install nginx-ingress stable/nginx-ingress --set
controller.replicaCount=1 --set controller.nodeSelector."beta\.kubernetes\.io/os"=linux
--set defaultBackend.nodeSelector."beta\.kubernetes\.io/os"=linux --set-string
controller.config.proxy-body-size=10m --set controller.service.externalTrafficPolicy=Local
```

In this example, `proxy-body-size` limits the size of payload requests, such as media data uploads or Sitecore packages installations, to 10 MB. You can adjust this setting to fit your installation.

For more information about configuring an ingress controller, see NGINX Ingress Controller Configuration.

3.2.5. Deploy the secrets

Sitecore Kubernetes deployments use secrets to securely store the strings that are used by the containers in the cluster.

To deploy the secrets:

1. In the `{topology}/secrets` folder, update all the secrets files – `.txt`, `.crt`, `.key` – with the appropriate values.
2. From the root folder, run the following command:

```
kubectl apply -k ./{topology}/secrets/
```

For more information about the secrets, see the Installation Guide for Production Environments with Kubernetes for the relevant Sitecore XP version.

3.2.6. Deploy an ingress configuration

- From the root folder, run the following command:

```
kubectl apply -k ./{topology}/ingress-nginx
```

3.2.7. Deploy the External Services for a non-production Deployment

1. From the root folder, run the following command:

```
kubectl apply -k ./{topology}/external/
```

2. To check the status of the pods, run the following command:

```
kubectl get pods -o wide
```

3. Wait until the status of all the pods is Running/OK.

```
kubectl wait --for=condition=Available deployments --all --timeout=900s
kubectl wait --for=condition=Ready pods --all
```

3.2.8. Deploy the data initialization jobs

1. From the root folder, run the following command:

```
kubectl apply -k ./overrides/{topology}/init/
```

2. To check the status of the jobs, run the following command:

```
kubectl get jobs -o wide
```

3. Wait until the status of all the jobs is Complete/OK.

```
kubectl wait --for=condition=Complete job.batch/solr-init --timeout=600s
kubectl wait --for=condition=Complete job.batch/mssql-init --timeout=600s
```

3.2.9. Deploy the Sitecore pods

To deploy the Sitecore pods:

1. From the root folder, run the following command:

```
kubectl apply -k ./overrides/{topology}/
```

2. To check the status of the pods, run the following command:

```
kubectl get pods -o wide
```

3. Wait until the status of all the pods is Running/OK.

```
kubectl wait --for=condition=Available deployments --all --timeout=1800s
```

3.2.10. Update the local host file

You must update the local host file with the external IP address of the ingress controller hostnames that are required by the ingress controller.

To obtain the external IP address of the ingress controller service for the CM role:

- Run this command:

```
kubectl get svc nginx-ingress-ingress-nginx-controller
```

The default hostnames are:

- cm.globalhost
- cd.globalhost
- id.globalhost

4. Post-deployment steps

When the deployment is finished, you must configure the SolrCloud search indexes.

IMPORTANT
You must perform the tasks described in this section if you have used Docker Compose or Kubernetes.

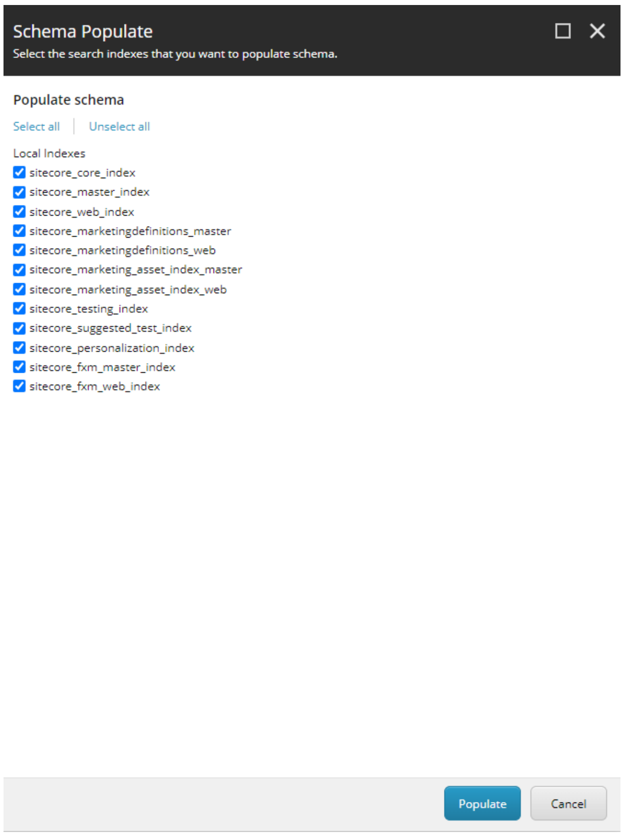
4.1. Configure the SolrCloud search indexes

To configure the SolrCloud search indexes:

1. Log in to the Sitecore content management instance or open a browser and navigate to:
    - Docker Compose - https://xp1cm.localhost/sitecore or https://xm1cm.localhost/sitecore
    - Kubernetes - https://cm.globalhost/sitecore

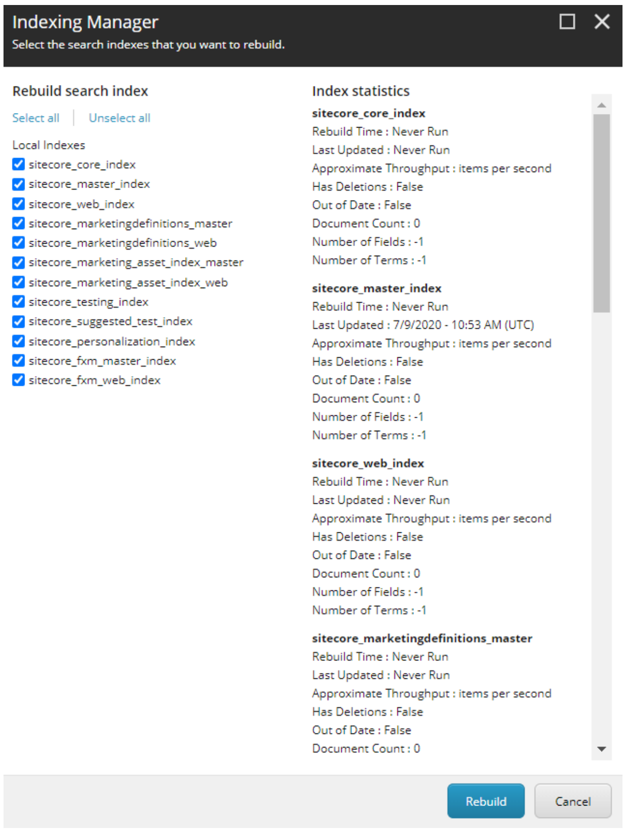
    NOTE
    You must log in to Sitecore with the *admin* user and password that you configured as a secret.

2. In the Sitecore Control Panel, click **Populate Managed Schema**. In the **Schema Populate** dialog box, select all the indexes and then click **Populate**.


Schema Populate
Select the search indexes that you want to populate schema.
Populate schema
Select all
Unselect all
Local Indexes
sitecore_core_index
sitecore_master_index
sitecore_web_index
sitecore_marketingdefinitions_master
sitecore_marketingdefinitions_web
sitecore_marketing_asset_index_master
sitecore_marketing_asset_index_web
sitecore_testing_index
sitecore_suggested_test_index
sitecore_personalization_index
sitecore_fxm_master_index
sitecore_fxm_web_index
Populate
Cancel

Wait for the process to complete. Close the dialog box.

3. In the Sitecore Control Panel, click **Indexing manager**. In the **Indexing Manager** dialog, select all the indexes that you want to rebuild and click **Rebuild**.



Indexing Manager
Select the search indexes that you want to rebuild.

Rebuild search index

Local Indexes
- [x] sitecore_core_index
- [x] sitecore_master_index
- [x] sitecore_web_index
- [x] sitecore_marketingdefinitions_master
- [x] sitecore_marketingdefinitions_web
- [x] sitecore_marketing_asset_index_master
- [x] sitecore_marketing_asset_index_web
- [x] sitecore_testing_index
- [x] sitecore_suggested_test_index
- [x] sitecore_personalization_index
- [x] sitecore_fxm_master_index
- [x] sitecore_fxm_web_index

Index statistics

sitecore_core_index
Rebuild Time : Never Run
Last Updated : Never Run
Approximate Throughput : items per second
Has Deletions : False
Out of Date : False
Document Count : 0
Number of Fields : -1
Number of Terms : -1

sitecore_master_index
Rebuild Time : Never Run
Last Updated : 7/9/2020 - 10:53 AM (UTC)
Approximate Throughput : items per second
Has Deletions : False
Out of Date : False
Document Count : 0
Number of Fields : -1
Number of Terms : -1

sitecore_web_index
Rebuild Time : Never Run
Last Updated : Never Run
Approximate Throughput : items per second
Has Deletions : False
Out of Date : False
Document Count : 0
Number of Fields : -1
Number of Terms : -1

sitecore_marketingdefinitions_master
Rebuild Time : Never Run
Last Updated : Never Run
Approximate Throughput : items per second
Has Deletions : False
Out of Date : False
Document Count : 0

Wait for the process to complete. Close the dialog box.