Sitecore Connect for Content Hub 4.0 Container Deployment Guide

A guide to deploying the Sitecore Connect for Content Hub connector on Docker and Azure Kubernetes Service.

December 14, 2021

Table of Contents

1. Introduction

The Sitecore Connect for Content Hub (SCCH) gives web editors and content/digital marketers the ability to work with content created in Sitecore Content Hub. When you make changes in Sitecore Content Hub, the connector automatically creates and updates content in Sitecore.

It also provides access to your digital assets in Sitecore DAM and allows you to easily embed them in your CMS solution.

For more information on Sitecore Content Hub see the Sitecore documentationthe Sitecore documentation.

This guide shows you how to add the SCCH connector to Sitecore container installations for Docker and Azure Kubernetes Service.

2. Prepare to deploy SCCH to Sitecore containers

This section explains what you need to deploy the Sitecore Connect for Content Hub (SCCH) connector to Sitecore containers for Docker and Azure Kubernetes Service.

2.1. Requirements

Before you add the SCCH module for Docker or AKS, you must have the following:

- Docker Desktop installed and running. For instructions on how to set up the Docker environment, see the Containers in Sitecore developmentContainers in Sitecore development documentation.
- If the installation is done on Docker, you must have the Sitecore Docker container files deployed on a local machine.. For instructions on how to prepare a Sitecore container, see the *Installation Guide for Developer Workstation with Containers* on the Sitecore download page.
- If the installation is done on Kubernetes, you must have the Sitecore AKS container files deployed on a local machine. For instructions on how to prepare a Sitecore environment with Kubernetes, see the *Installation Guide for Production Environment with Kubernetes* on the Sitecore download page.
- You must have installed the Sitecore certificates, and added the Sitecore domain name to your host file in Windows.

3. Add the SCCH connector module to Sitecore in Docker

To add Sitecore Connect for Content Hub (SCCH) in Docker, you must do the following in this order:

- Prepare the installation files.
- Build the Docker images.
- Update the Solr indexes.

3.1. Prepare the installation files

To prepare the files you need for the installation:

1. Download the SCCH container deployment package from the Sitecore download page. Extract it to your local workstation with the folder structure intact.
2. Download the Sitecore Experience Platform container deployment package from the Sitecore download page. Extract it to a new folder on your local workstation with the folder structure intact. Name the new folder, for example, `ContentHub`.
3. In the folder where you extracted the SCCH package in Step 1, navigate to the folder for the Windows version and topology you are using, for example, `Chub.Asset\compose\ltsc2019\xp1`. Copy the `docker-compose.override.yml` file to the `ContentHub\compose\<version>\<topology>` folder you created in Step 2.
4. To build a Content Hub image on top of your Sitecore images, open the `ContentHub\compose\<version>\<topology>\docker-compose.override.yml` file and add a `build` section for the image.
   To build, for example, a `cm` image, add the following:

```
cm:
 build:
  context: ./build/cm
  args:
      BASE_IMAGE: <insert sitecore base image here>
      CH_IMAGE: ${CH_IMAGE}
      TOOL_IMAGE: ${TOOL_IMAGE}
      SITECORE_ROLE: cm
      SITECORE_ROOT: .\
```

5. If you are deploying to an XP1 or XM1 topology, you must add a `cd` image. To do so add, for example, the following:

```
cd:
build:
 context: ./build/cd
 args:
    BASE_IMAGE: ${SITECORE_DOCKER_REGISTRY}sitecore-${TOPOLOGY}-cd:${SITECORE_VERSION}
    CH_IMAGE: ${CH_IMAGE}
```

```
    TOOL_IMAGE: ${TOOL_IMAGE}
    SITECORE_ROLE: cm
    SITECORE_ROOT: .\
```

6. If you are deploying to Kubernetes, you must also add a `build` section for the `mssql-init` image. For example:

```
mssql-init:
 image: <mssql-init image name>
 build:
  context: ./build/mssql-init
  args:
      BASE_IMAGE: <insert sitecore base image here>
      CH_IMAGE: ${CH_IMAGE}
```

7. In the `ContentHub` folder, navigate to the folder for your Windows version and topology you are using, for example, `compose\ltsc2019\xp1`. Open the `.env` file in a text editor. At the bottom of the file, add a definition for the CH image. For example:

```
CH_IMAGE= scr.sitecore.com/sxp/modules/sitecore-chub-<topology-to-deploy>-assets:4.0.0-
<target-OS-to-deploy>
```

8. In the folder where you extracted the SCCH package, open the `.env-example` file in a text editor, and copy all the contents.
9. At the bottom of the `.env` file from Step 7, after the CH image data, paste in the contents from the `.env-example` file. Save the `.env` file.

3.2. Build the Docker images

When you have prepared the installation files, you must create Docker files for each role, and build the Docker images.

> **NOTE**
> For more information on image assets, see the documentation on how to Add Sitecore ModulesAdd Sitecore Modules.

To build the images:

1. In the `ContentHub` folder, navigate to the folder for your Windows version and topology you are using, for example, `compose\ltsc2019\xp1`. Create a subfolder and name it `build`.
2. In the `build` folder, create these subfolders:
   - `mssql`
   - `mssql-init`

> **NOTE**
> You only need the `mssql-init` subfolder if you are deploying to Kubernetes.

- `cm`
- `cd`

> **NOTE**
> You only need the `cd` subfolder if you are deploying to an XP1 or XM1 topology.

3. In each subfolder, create a new file and name it `Dockerfile`.
4. In the `mssql` folder, in the `Dockerfile` file, enter the following instructions:

```
# escape=`
ARG BASE_IMAGE
ARG CH_IMAGE
FROM ${CH_IMAGE} as ch-connector
FROM ${BASE_IMAGE}
SHELL ["powershell", "-Command", "$ErrorActionPreference = 'Stop'; $ProgressPreference =
'SilentlyContinue';"]
# Deploy CH connector dacpac file
COPY --from=ch-connector \module\db \ch_data
RUN C:\DeployDatabases.ps1 -ResourcesDirectory C:\ch_data; `
    Remove-Item -Path C:\ch_data -Recurse -Force;
```

5. If you are deploying to Kubernetes, in the `mssql-init` folder, in the `Dockerfile` file, enter the following instructions:

```
# escape=`
ARG BASE_IMAGE
ARG CH_IMAGE
FROM ${CH_IMAGE} as ch-connector
FROM ${BASE_IMAGE}
SHELL ["powershell", "-Command", "$ErrorActionPreference = 'Stop'; $ProgressPreference =
'SilentlyContinue';"]
FROM ${BASE_IMAGE} AS ch
COPY --from=ch-connector C:\module\db C:\resources\ch
```

6. In the `cm` folder, in the `Dockerfile` file, enter the following instructions:

```
# escape=`
ARG BASE_IMAGE
ARG CH_IMAGE
ARG TOOL_IMAGE
FROM ${CH_IMAGE} as def
FROM ${TOOL_IMAGE} as tooling
FROM ${BASE_IMAGE}
ARG SITECORE_ROLE
ARG SITECORE_ROOT
SHELL ["powershell", "-Command", "$ErrorActionPreference = 'Stop'; $ProgressPreference =
'SilentlyContinue';"]
WORKDIR C:\inetpub\wwwroot
# Add CH connector module
COPY --from=ch-connector "\module\${SITECORE_ROLE}\content" ${SITECORE_ROOT}
#Copy transformation files
COPY --from=ch-connector \module\transforms\C:\transforms\
#Add tools from sitecore-docker-tools-assets
COPY --from=tooling \tools\C:\tools\
RUN C:\tools\scripts\Invoke-XdtTransform.ps1 -Path C:\inetpub\wwwroot -XdtPath \transforms
\cm
```

7. If you are deploying to an XP1 or XM1 topology, in the `cd` folder, in the `Dockerfile` file, enter the following instructions:

```
# escape=`
ARG BASE_IMAGE
ARG CH_IMAGE
ARG TOOL_IMAGE
FROM ${CH_IMAGE} as ch-connector
FROM ${TOOL_IMAGE} as tooling
FROM ${BASE_IMAGE}
ARG SITECORE_ROLE
ARG SITECORE_ROOT
SHELL ["powershell", "-Command", "$ErrorActionPreference = 'Stop'; $ProgressPreference =
'SilentlyContinue';"]
WORKDIR C:\inetpub\wwwroot
# Add CH connector module
COPY --from=ch-connector "\module\${SITECORE_ROLE}\content" ${SITECORE_ROOT}
#Copy transformation files
COPY --from=ch-connector \module\transforms\C:\transforms\
#Add tools from sitecore-docker-tools-assets
COPY --from=tooling \tools\C:\tools\
RUN C:\tools\scripts\Invoke-XdtTransform.ps1 -Path C:\inetpub\wwwroot -XdtPath \transforms
\cd
```

8. In the Windows console, go to the `ContentHub\compose\<version>\<topology>` folder where the `docker-compose.override.yml` file is. Run the command `docker-compose build`.
9. After the build has completed, run the command `docker-compose up -d`.

> **NOTE**
> Some modifications to Sitecore deployments, such as adding connection strings or changing the web configuration files, require you to use configuration transforms to change the configuration files. For information on how to apply configuration transforms, see the Sitecore container development documentationcontainer development documentation.

When the Docker compose command has finished, update your Solr indexes.

4. Add the SCCH connector module to Sitecore in Azure Kubernetes Service

To add the Sitecore Connect for Content Hub (SCCH) connector in Azure Kubernetes Service (AKS) you must do the following in this order:

- Build the SCCH images and push them to Azure.
- Prepare files and folders for deployment.
- Deploy the containers using *kubectl* commands.
- Update your Solr indexes.

4.1. Build images and push them to Azure

To build the images for SCCH and push them to Azure:

1. Build the images for SCCH as explained in Add the SCCH connector module to Sitecore in Docker.
2. Open the Windows console and use the `docker tag` command to tag the images. For example:

```
docker tag sitecore-chub-xp1-assets:<tag version> $registry/sitecore-chub-xp1-cm:<tag version>
```

3. In the console, use the `docker push` command to push the images to your Azure registry. For example:

```
docker push $registry/sitecore-chub-xp1-cm:<tag version>
```

4.2. Prepare files and folders for deployment

To prepare files and folders in your installation for deployment:

1. In the folder where you extracted the SCCH container deployment package, navigate to the `k8s\<version>` folder, for example, `k8s\ltsc2019`. Copy the `overrides` subfolder to the Sitecore Experience Platform (SXP) container deployment package folder `k8s\<version>` (on the same level as the `xp1` folder).
2. In the SXP container deployment package, in each of the `overrides`, `overrides\<topology>\init`, and `overrides\<topology>\secrets` folders, locate the

`kustomization.yaml` file. In each file, update the `bases` parameter with the appropriate folder names for your installation, for example, `../../xp1`.

> **NOTE**
> The `bases` parameter contains the placement of the original Sitecore container deployment files that the `kustomization.yaml` files override.

3. In the `kustomization.yaml` files, in the `images:` section, update the `newName` and `newTag` parameters with the values for the images you pushed to the Azure Registry, for example, `mssql-init`, `cm`, and `cd`.

> **NOTE**
> You only need to update parameters for a `cd` image, if you are deploying to an XP1 or XM1 topology.

4. In the `overrides\<topology>\secrets` folder, update each of the following files with connection string details:
   - `sitecore-cmp-content-hub.txt`
   - `sitecore-cmp-service-bus-entity-path-in.txt`
   - `sitecore-cmp-service-bus-entity-path-out.txt`
   - `sitecore-cmp-service-bus-subscription.txt`
   - `sitecore-dam-content-hub.txt`
   - `sitecore-dam-external-redirect-key.txt`
   - `sitecore-dam-search-page.txt`

> **NOTE**
> Each file contains an example of how the connection string should look.

4.3. Deploy the containers

Prepare the AKS cluster configuration and deploy the ingress controller. For information on how to do this, see the *Installation Guide for Production Environment for Kubernetes*, which is available on the Sitecore download page.

To deploy the containers and the necessary Kubernetes components:

1. Open the Windows console, and navigate to the `k8s\<version>` folder.
2. Deploy the secrets. Use this command:

```
kubectl apply -k ./overrides/<topology>/secrets/
```

3. Run the `external` folder. Use this command:

```
kubectl apply -k ./<topology>/external/
```

4. Wait for all containers to have the status *Ok/Running*. You can check the status with this command:

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

5. Run the `init` folder. Use this command:

```
kubectl apply -k ./overrides/<topology>/init/
```

6. Wait for all containers to have the status *Completed*. You can check the status with this command:

```
kubectl get pods
```

7. If you are using the `XP1` topology, and you want to create persistent volumes, run this command:

```
kubectl apply -f ./xp1/volumes/azurefile
```

> **NOTE**
> Persistent volumes are only available for the `XP1` topology

8. Run the Sitecore containers with the SCCH changes. Use this command:

```
kubectl apply -k ./overrides/<topology>/
```

9. Wait for all containers to have the status *Ok/Running*. You can check the status with the `kubectl get pods` command.
10. In order to enable external access, you need to update the local host file with the external IP address. To obtain the external IP address, use this command:

```
kubectl get service -l app=nginx-ingress
```

> **NOTE**
> For information on how to update the local host file, see the *Installation Guide for Production Environment for Kubernetes*, which is available on the Sitecore download page.

When the containers have been deployed, update your Solr indexes.

5. Update the Solr indexes

When you have deployed the containers, you must update your Solr indexes.

To update the indexes:

1. Browse to your Sitecore URL, for example, `https://xp1cm.localhost/`. Open the control panel and click **Populate Solr Managed Schema**.
2. After Sitecore has populated the Solr Schema, click **Indexing Manager**.
3. In the **Indexing Manager** dialog, select the following indexes:
    - sitecore_master_index
    - sitecore_core_index
    - sitecore_web_index
    - sitecore_marketing_asset_index_master
    - sitecore_marketing_asset_index_web databases
4. Click **Rebuild**. When the indexes have been rebuilt, click **Close**.
5. Open the Content Editor with *Master* as the content database.
6. In the content tree, navigate to */sitecore/system/Data Exchange*. On the **Folder** tab, verify that the **Empty Data Exchange Tenant** and **Connect for SFMC Tenant** buttons are available.