SITECORE

Kubernetes deployment guide for Sitecore Horizon 10.1.1

A guide to deploying Horizon in a Kubernetes environment.

Table of Contents

1. Deploy Horizon to Kubernetes

Introduction to the guide for deploying Horizon to a Sitecore installation using Kubernetes.

This guide explains how to deploy Horizon to a Sitecore container installation using Kubernetes.

> **NOTE**
> In the examples in this guide, replace <*topology*> with *xp1* or *xm1*, depending on which Sitecore topology you are deploying.

1.1. Prerequisites for deploying Horizon to Kubernetes

Guide to preparing for deploying Horizon to a Kubernetes installation for Sitecore.

In order to get ready to deploy Horizon to Kubernetes, you must prepare the Kubernetes and Sitecore configuration files, make sure you have a working Kubernetes installation for Sitecore, and install the requisite TLS/SSL certificates. You can perform these actions in any order.

1.1.1. Prepare Kubernetes specification files

To obtain the specification files for Kubernetes for your setup:

1. In the GitHub Sitecore containers deployment repository, locate and download the Sitecore Horizon Container Deployment zip package for 10.1.1. Unzip it and locate the `k8s\ltsc2019` folder. Copy this folder to your local work folder.
2. Download the Sitecore Experience Platform 10.1.1 Container Deployment Package from the Sitecore download page. Extract the archive and locate the `k8s\ltsc2019\<topology>` folder for the Sitecore topology that you want to deploy, for example, `k8s\ltsc2019\xp1`.
3. Copy the `<topology>` folder, for example, *xp1*, to your working folder next to the Horizon configuration.

For example, for the *xp1* topology, you now have the following folder structure in your work folder:

- *horizon*
- *overrides*
- *xp1*

1.1.2. Prepare your Sitecore Kubernetes platform

Download the *Installation Guide for Production Environments with Kubernetes for Sitecore 10.1.1* and apply all steps from the *Prerequisites* section to your local Sitecore Kubernetes configuration.

1.1.3. Generate certificates

The Horizon module installs a runnable service which external users must be able to access. In order to make it accessible, you must install the TLS/SSL certificates that the NGINX ingress controller requires.

To generate the TLS/SSL certificates:

1. Open a Windows Command Prompt as an Administrator.
2. Navigate to the `overrides/<topology>` folder.
3. Run 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 *.crt
del /Q /S *.key
mkcert -cert-file secrets\tls\global-hrz\tls.crt -key-file secrets\tls\global-hrz\tls.key
"hrz.globalhost"
```

> **NOTE**
> The first time the mkcert utility runs, it prompts the user to install the generated self-signed root certificate authority.

1.2. Build images for Horizon on a Sitecore container platform

How to build custom content management and MSSQL-INIT images for use with Horizon on a Sitecore container platform.

In order to prepare the Sitecore platform images to work with the Horizon service, you must install the Horizon integration asset image on top of Sitecore content management (CM) and MSSQL-INIT images.

1.2.1. Build custom images

To build the custom CM and MSSQL-INIT images:

1. On your Docker installation, create an empty *build* folder.
2. In the *build* folder, create a new folder and name it *cm*. In the *cm* folder, create a new file and name it `Dockerfile`. Copy and paste the following into the new file:

```
# escape=`
ARG BASE_IMAGE
ARG HORIZON_INTEGRATION_ASSET_IMAGE
FROM ${HORIZON_INTEGRATION_ASSET_IMAGE} as horizon_integration
FROM ${BASE_IMAGE} AS base
COPY --from=horizon_integration C:\module\cm\content C:\inetpub\wwwroot
```

3. In the *build* folder, create a new folder and name it *mssql-init*. In the *mssql-init* folder, create a new file, and name it `Dockerfile`. Copy and paste the following into the new file:

```
# escape=`
ARG BASE_IMAGE
ARG HORIZON_INTEGRATION_ASSET_IMAGE
FROM ${HORIZON_INTEGRATION_ASSET_IMAGE} as horizon_integration
```

```
FROM ${BASE_IMAGE} AS cm
COPY --from=horizon_integration C:\module\db C:\resources\hrz
```

4. In the *build* folder, create an empty file and name it `docker-compose.yml`. Copy and paste the following into the new file:

```
version: "3.7"
services:
  cm:
    image: sitecore-horizon-integration-${TOPOLOGY}-cm:${VERSION}
    build:
      context: ./cm
      args:
        BASE_IMAGE: ${CM_BASE_IMAGE}
        HORIZON_INTEGRATION_ASSET_IMAGE : ${HORIZON_INTEGRATION_ASSET_IMAGE}
  mssql-init:
    image: sitecore-horizon-integration-${TOPOLOGY}-mssql-init:${VERSION}
    build:
      context: ./mssql-init
      args:
        BASE_IMAGE: ${MSSQL_INIT_IMAGE}
        HORIZON_INTEGRATION_ASSET_IMAGE : ${HORIZON_INTEGRATION_ASSET_IMAGE}
```

5. In the *build* folder, create a file and name it `.env`. Populate the file with the following parameters:

Variable	Description
HORIZON_INTEGRATION_ASSET_IMAGE	The Horizon integration asset image. The name of the image is `scr.sitecore/sxp/modules/horizon-integration-<topology>-assets:2.26-ltsc2019`. Replace <topology> with the name of the Sitecore topology you are using, either *xp1* or *xm1*.
CM_BASE_IMAGE	Sitecore CM image for each Sitecore CMS build. You can find this value in the Sitecore platform `docker-compose.yml` and `.env` files.
TOPOLOGY	The topology you want to deploy Horizon to. It can be *xp1* or *xm1*.
MSSQL_INIT_IMAGE	Sitecore MSSQL-INIT image for each Sitecore CMS build. Look for this data in the Sitecore platform `docker-compose.yml` and `.env` file
VERSION	The version tag for the newly generated CM and MSSQL-INIT images patched with Horizon integration data, for example, *10.1.1*.

6. In the *build* folder, run the `docker-compose build` command. This builds CM and MSSQL-INIT images patched with Horizon files.
7. Push the CM and MSSQL-INIT images to a private container registry, so the deployment process can pick them up later. See the Docker CLI documentation for information on how to use the `docker push` command.

1.2.2. Configure Images

You must configure the patched CM and MSSQL-INIT images in the Horizon Kubernetes configuration files.

To configure the images:

1. Open the `overrides\<topology>\kustomization.yaml` file. Change the values of the *newname* and *newTag* properties to the corresponding image name and tag for the CM image you created.
2. Open the `overrides\<topology>\init\kustomization.yaml` file. Change the values of the *newName* and *newTag* properties to the corresponding image name and tag for MSSQL-INIT image you created.

1.3. Enable SXA and ContentHub DAM integration

How to enable SXA and Content Hub DAM modules in Horizon on a Sitecore container platform.

If you are deploying the Sitecore XP with SXA and ContentHub DAM modules, and you want to use them with Horizon, you must enable support for them.

To enable support for SXA and ContentHub DAM in Horizon:

1. Open the `horizon\kustomization.yaml` file.
2. Add the following section to the end of the file:

```
patchesStrategicMerge:
  - patch-hrz-enable-dam.yaml
  - patch-hrz-enable-sxa.yaml
```

3. If you want to enable a single module support, SXA or ContentHub DAM, add only the patch line for the module you want to enable. For example, to enable just SXA:

```
patchesStrategicMerge:
  - patch-hrz-enable-sxa.yaml
```

1.4. Walkthrough: deploying the Sitecore platform with Horizon to an Azure Kubernetes Server

How to deploy a Sitecore platform with Horizon to an Azure Kubernetes server using Kubectl CLI commands.

This walkthrough describes how to deploy Sitecore Experience Platform with Horizon to the Azure Kubernetes Service.

You use the Kubect CLI (Command Line Interface) to deploy the Sitecore and Horizon containers to a Kubernetes cluster.

To deploy the containers, you must:

- Create an AKS cluster

- Configure the Kubectl context cluster
- Configure access to a private container registry
- Deploy an ingress configuration
- Deploy the secrets
- Deploy an ingress controller
- Deploy External Services for a non-production deployment
- Deploy the data initialization jobs
- Deploy the Sitecore pods
- Update the local host file
- Configure the SolrCloud search indexes

1.4.1. Create an AKS cluster

To create a new Azure Kubernetes Service (AKS) cluster with a Windows Server 2019 node pool, 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 documentationAzure AKS documentation.

1.4.2. Configure the Kubectl context cluster

To 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
```

1.4.3. Configure access to a private container registry

The deployed Kubernetes cluster requires access to the private container registry to which you have pushed your patched *cm* and *mssql-init* images. To let a Kubernetes cluster authenticate with a private container registry and pull images from it, you must create an image pull secret. The secret name must be *sitecore-docker-registry*.

1.4.4. Deploy an ingress configuration

To deploy an ingress configuration:

- From the root folder, run this command:

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

1.4.5. Deploy the secrets

To deploy the secrets:

1. Ensure that all the secrets files (.txt, .crt, .key) files in the `<topology>/secrets` and `overrides\<topology>\secrets` folders are updated with proper values.
2. From the root folder, run this command:

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

1.4.6. Deploy an ingress controller

To deploy an ingress controller:

1. Use the Windows AMD64 binaries to Install Helm. You can also use an alternative method as described in Installing Helm Through Package Managers.
2. Add an NGINX ingress controller feed to Helm. For example:

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

3. Update the feed using the command `helm repo update`.
4. Use Helm to deploy the NGINX ingress controller. For example:

```
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
```

> **NOTE**
> In the example, *proxy-body-size* limits payload requests, such as media data upload or Sitecore packages installation, to 10 MB. You can adjust this to fit your installation. For more information about ingress configuration, see NGINX Ingress Controller Configuration.

1.4.7. Deploy External Services for a non-production deployment

To deploy External Services for a non-production deployment:

1. From the root folder, run this command:

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

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

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

3. To wait until the status of all the pods is Running/OK, run this command:

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

1.4.8. Deploy the data initialization jobs

To deploy the data initialization jobs:

1. From the root folder, run this command:

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

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

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

3. To wait until the status of all the jobs is Complete/OK, run this command:

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

1.4.9. Deploy the Sitecore pods

To deploy the Sitecore pods:

1. From the root folder, run this command:

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

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

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

3. To wait until the status of all the pods is Running/OK, run this command:

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

1.4.10. Update the local host file

To update the local host file:

1. To obtain the external IP address of the ingress controller service for the CM role, run this command:

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

2. Update the local host file with the external IP address and the hostnames that are required by the ingress controller. The default hostnames are:
   - `cm.globalhost`
   - `cd.globalhost`
   - `Id.globalhost`
   - `hrz.globalhost`

1.4.11. Configure the SolrCloud search indexes

To configure the SolrCloud search indexes:

1. Open a browser and navigate to https://cm.globalhost/sitecore.
2. Log in to Sitecore with the admin user and password that you configured as a secret.
3. In the Sitecore Control Panel click **Populate Managed Schema**. In the **Schema Populate** dialog box, select all the indexes and click **Populate**.

4. In the Sitecore Control Panel, in the **Indexing** section, click **Indexing Manager**.
5. In the **Indexing Manager** dialog box, select **sitecore_horizon_index** and then click **Rebuild**.
6. When the search indexes have been rebuilt, click **Close**.