Sitecore Upgrade Container Deployment Guide

Sitecore XP 10.2.0

Upgrade SQL Server databases to Sitecore XP 10.2.0

Table of Contents

Chapter 1

Introduction

Since Sitecore Experience Platform 10.0.1, you can use a Sitecore upgrade container to upgrade Sitecore SQL databases. The Sitecore upgrade container upgrades Sitecore XP and xConnect databases that are stored on-prem in containers, or in an Azure SQL environment. You can run the Sitecore upgrade container in Docker Compose and in Kubernetes environments.

With Sitecore XP containers, you can upgrade your Sitecore solution with almost no downtime. You can use the upgrade container to upgrade both single-instance and multi-instance installations.

The upgrade process for Sitecore XP 10.2.0 supports database upgrades from the Sitecore XP 9.3.0 or later to Sitecore XP 10.2.X.

This chapter contains the following sections:

- Sitecore Container Deployment Package
- Software requirements
- Prerequisites
- Limitations
- Compatibility

1.1 Sitecore Container Deployment Package

The Sitecore Container Deployment Package is a set of tools that you use to upgrade the SQL Server databases of your existing Sitecore XP 9.3.0 - 10.1.X installation to Sitecore XP 10.2.0.

These tools use containers to upgrade the SQL databases and can be used in both Docker Compose and Kubernetes environments. You can use these tools to upgrade both standard Sitecore installations and containerized installations to either a standard Sitecore installation or to a containerized installation. However, these tools are primarily designed to upgrade containerized installations.

You can use this package to update both XP and XM topologies.

You must edit the tools in the package to provide the relevant information about your Sitecore solution.

When you run the tools, they download the Sitecore upgrade container image from the Sitecore Container Registry and deploy this container to upgrade the SQL databases of your existing Sitecore XP installation.

Since Sitecore XP 10.0.1, the Sitecore upgrade container contains all the required information about the changes that have been implemented in Sitecore.

The Sitecore container upgrade assets are:

- Sitecore SQL upgrade Docker image

  A Docker image with all the tools required to upgrade Sitecore SQL databases.
- Docker Compose configuration files.
- Kubernetes specification files.

1.1.1 Topologies

You can use the Sitecore container upgrade assets to upgrade Sitecore SQL databases in the following topologies:

- XP Scaled (XP 1)

  To upgrade the Sitecore Experience Platform databases, use the sitecore-xp1 upgrade container assets either for Docker Compose or Kubernetes.
- XM Scaled (XM 1)

  To upgrade the Sitecore Experience Manager databases, use the sitecore-xm1 upgrade container assets either for Docker Compose or Kubernetes.

1.1.2 Sitecore upgrade Docker Compose configuration files

The Sitecore upgrade Docker Compose files for the Sitecore XP and Sitecore XM topologies are:

- `Docker-compose.upgrade.yml`

  The Docker Compose configuration file that contains information about the upgrade containers and its configuration.
- `upgrade.env`

  The Docker environment file that declares the default environment variables used in the compose file and by the upgrade container.

For more information about which environment variables are used in the Docker Compose environments for the supported topologies, see Appendix C – The environment variables list .

1.1.3 Sitecore upgrade Kubernetes specification files

The Sitecore Kubernetes specification files for the Sitecore XP and Sitecore XM topologies are:

- `mssql-upgrade.yaml`

  The Kubernetes specification file that you use to deploy, configure, and run the upgrade container in a Kubernetes cluster.

- `kustomization.yaml`

  The file that deploys all the secret names and values required by the upgrade container.

- Secrets and ConfigMap values

  The text files used to store values for Kubernetes secrets and the ConfigMap values used by the upgrade container. The files are stored in Kubernetes specification files for each topology in the `/configuration/` folder.

For a complete list of the secrets and ConfigMap values as well as more information about them, see Appendix B – The Kubernetes secrets list .

1.2 Software requirements

1.2.1 Docker host machine software requirements

If you decide to run Sitecore container upgrade in a Docker Compose environment, your host machine should meet the following requirements:

- Operating System:
  - Windows 10 1809 or later

    or
  - Windows Server 1809 or later
- Docker Desktop for Windows

You must also download:

- Sitecore Container Deployment Package

  `SitecoreContainerDeployment.10.2.0.006766.X683.zip`

  Extract the `\compose\[Your Windows Server version]\upgrade\xm(p)1` configuration folder for the desired topology.

1.2.2 Kubernetes cluster and client software requirements

If you decide to use Kubernetes to run the container upgrade, your environment must meet the following requirements:

Kubernetes cluster requirements:

- Kubernetes 1.16.x or later
- Windows Server 2019 version 1809

Client software requirements:

- Operating System
  - Windows 10 1809 or later

    or
  - Windows Server 1809 or later
- Kubectl 1.16x or later

  Use the latest stable non-preview version.

  To get a list of the supported locations run the following command:

  ```
  az account list-locations
  ```

  To get the latest stable version with the desired region (location) run the following command:

  ```
  az aks get-versions --location <location> --output table
  ```
- Azure CLI 2.8.0 or later is required for AKS deployments.
- Sitecore Container Deployment Package

  `SitecoreContainerDeployment.10.2.0.006766.683.zip`

Extract the `\k8s\ltsc2019\upgrade\xm(p)1` configuration folder for the topology that you want to deploy.

1.3 Prerequisites

You can use the Sitecore container upgrade procedure to upgrade your Sitecore XP installation if your environment fulfills the following prerequisites:

- Sitecore XP 9.3.0 - 10.1.X deployed on a container cluster or to a computing infrastructure that uses the appropriate state configuration – Kubernetes, Docker or WDP/SAT/SIF.
- The source code of your installation can produce the relevant deployment and content packages – Docker images or WDP packages.
- You have created backups of your Sitecore databases.

  Important
  We strongly recommend that you upgrade copies of your production databases that contain all your data.

- The Sitecore databases should not have any active connections from the Web roles or from the xConnect worker roles.
- The upgrade assets must be able to access the SQL Server instance that hosts the Sitecore database over the Internet, a private domain network, or from a virtual Docker or Kubernetes network.
- There is a SQL Server user with permission to update the Sitecore database schemas and run SQL scripts against them, for example, the admin user.

1.4 Limitations

This document describes how to upgrade of a single Sitecore XP environment. You must repeat the procedure for each environment and for each version.

The database upgrade process is irreversible. Before you upgrade a solution, we recommend that you verify the upgrade process in an isolated environment.

The upgrade process describes how to upgrade Sitecore databases located on a single SQL Server instance.

For information about the configuration and filesystem upgrade of individual roles and the required post-upgrade steps, see the Upgrade Guide for Sitecore XP 10.2.0.

1.5 Compatibility

The Sitecore container upgrade artifacts described in this document can only be used to upgrade from Sitecore XP 9.3.0 - 10.1.X to Sitecore XP 10.2.0 and only cover the Sitecore XP and xConnect SQL databases.

The upgrade process described here does not apply to databases deployed on other database providers, such as MongoDB. After you upgrade Sitecore XP, you must upgrade all the modules separately.

For more information about upgrading the SQL Server session state provider, see to the Sitecore 10.2.0 Upgrade Guide.

Chapter 2

Upgrade

When the requirements and prerequisites are in place, you can start the upgrade process.

To upgrade installed modules, see the section Modules upgrade

This chapter contains the following sections:

- Prepare for the upgrade
- The connection strings to databases
- Deployment
- Sitecore SQL Upgrade process
- Troubleshooting
- Clean up
- Modules upgrade
- Upgrade Sitecore Identity Server

2.1 Prepare for the upgrade

To run the Sitecore container upgrade process, you must prepare the following requirements:

- Download and extract the Sitecore Container Deployment package.

  This package contains the Sitecore upgrade Docker Compose files and Sitecore Kubernetes specification files for each Sitecore topology.
- Specify the environment variables.
- Prepare the Sitecore license file.

2.1.1 The environment variables

The environment variables are the preferred mechanism for passing the configuration settings into the Sitecore upgrade container.

The environment variables in Docker Compose

The environment variables for Docker Compose are stored in the environment variable configuration file – `upgrade.env`. Docker Compose loads these variables automatically during startup.

Before you deploy the Sitecore upgrade container, you must specify all the environment variables with the required values.

For a complete list of Docker env variables and more information about the individual variables, see Appendix C – The environment variables list

Important
All the environment variables must fit in the `upgrade.env` file, in a single 32,767character block. If the total size of the variables exceeds this size, you will be unable to set new values and the databases will not be upgraded successfully.

To reuse environment variables across multiple environments, you should consider setting environment variables on the host Windows OS and removing the corresponding keys from the environment variable configuration file used by Docker Compose.

Kubernetes configuration

The Sitecore upgrade Kubernetes deployment uses secrets to securely store the strings that are used by the container in the Kubernetes cluster.

The secrets are used to store the SQL Server user name and password, the Sitecore license, and so on.

The Kubernetes configuraton also includes ConfigMap values such as the SQL server address and the prefix of the Sitecore databases which are required for the upgrade container.

Before you deploy the Sitecore upgrade container to the Kubernetes cluster, you must update the secrets and ConfigMap values in the configuration text files with the required values.

For a complete list of the secrets and the ConfigMap values see Appendix B – The Kubernetes secrets list

We provide a Kubectl `kustomization.yaml` file that deploys all the secrets and the ConfigMap values in a single command.

The Sitecore license file

The Sitecore license file is typically passed to the container instances as an environment variable in encoded string form. The Sitecore license file is very large. You must therefore compress and Base64 encode it to ensure that it conforms 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 Docker environment variable configuration file or copy the string value to the license secret text file – `sitecore-license.txt` – depending on which orchestrator you choose for the upgrade container.

Appendix A – License file compression and encoding PowerShell helper function contains a sample PowerShell script that converts a license file into a Base64 compressed string for use in an environment variable.

2.2 The connection strings to databases

In the container upgrade process we use connection strings to connect to the SQL databases that must be upgraded. These connection strings are listed in the Docker Compose configuration file – `docker-compose.upgrade.yml` and in the Kubernetes specification file – `mssql-upgrade.yaml`.

All the database names in the connection strings are resolved with the *sitecore* prefix by default, for example, *sitecore.Core*. If your databases use another prefix you can specify it in the corresponding environment variable or Kubernetes secret. If you use custom database names which do not follow the *[prefix].[database name]* convention, you must specify the correct database names in the Docker Compose or Kubernetes specification files.

If the Kubernetes specification file contains the connection string for a database, the upgrade script and Sitecore item update procedure will be executed for that database.

If you do not want to upgrade some databases, you must delete their connection strings from the specification file.

The Kubernetes specification files for the upgrade container have connection strings for only two xDB Collection shards by default. If your solution contains more *xDB Collection* database shards, you must add a connection strings for them by using the same format that is used for the other shards.

Note
If you add connection strings for shard databases, you must increment the shard number in the connection string name, for example, shard 3 should be `Sitecore_ConnectionStrings_Xdb_Collection_Shard3`.

Note
The Sitecore upgrade container does not support MongoDB databases. If your xDB Collection database is stored in MongoDB you must remove the `Sitecore_ConnectionStrings_Xdb_Collection_Shard*` connection strings from the specification files for the upgrade container.

2.3 Deployment

The Sitecore Container Deployment Package contains two sets of the upgrade container artifacts – one for Docker Compose and one for Kubernetes.

2.3.1 Docker Compose deployment

To deploy the Sitecore upgrade container in Docker Compose:

1. In Docker for Windows, switch to Windows container mode.
2. Download and extract the Sitecore Container Support 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 Support Package to and open the folder with the Docker Compose upgrade artifacts for your Windows version and the Sitecore topology that you want to upgrade, for example, `\compose\ltsc2019\upgrade\xp1`.
4. In the topology folder, run the `compose-init.ps1` script.

   This script updates the environment configuration file with the appropriate values for all the environment variables including the SQL user name, SQL password, SQL Server address, and the Sitecore license file.

   For more information about running the script to prepare for the deployment, see the Installation Guide for Developer Workstation with Containers.
5. Open the `docker-compose.upgrade.yml` file.

   Study this file to get a better understanding of the container and connection strings needed to upgrade the databases
6. Update the connection strings as required.
7. In the Windows console, go to the folder that contains the `docker-compose.upgrade.yml` file and run the following Docker Compose command:

   ```
   docker-compose.exe -f .\docker-compose.upgrade.yml --env-file .\upgrade.env up
   ```

   The `--env-file` parameter was introduced in Docker Compose 1.25.0.

   Docker Compose pulls the mssql-upgrade image from the Sitecore Container Registry and deploys the container which immediately starts the upgrade process.

   When the upgrade process is completed, the upgrade container stops.
8. To check the status of the Docker container, run the following command:

   ```
   docker-compose.exe -f .\docker-compose.upgrade.yml --env-file .\upgrade.env ps
   ```

   This command provides the current status of the upgrade container.

2.3.2 Kubernetes deployment

To deploy the Sitecore upgrade container to the Azure Kubernetes Service, you must have a Kubernetes 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 and the cluster must have access to the Docker registry with the upgrade images.

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

Configure the kubectl context cluster

Kubectl is the Kubernetes command-line tool and allows you to run commands against Kubernetes clusters.

To configure the kubectl context cluster

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

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

2. Get the credentials for the Kubernetes cluster that was created with the AKS cluster.

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

Deploy the Sitecore upgrade job

To deploy the Sitecore upgrade job:

1. Ensure that all the secrets files (.txt) in the `./configuration` folder are updated according to the requirements listed in *Appendix B – The Kubernetes secrets* list.
2. From the root folder of the desired topology, run the following command:

```
kubectl apply -k .\
```

   This command:

   - Deploys the secrets.
   - Populates the ConfigMap values.
   - Pulls the Sitecore upgrade container image from the Sitecore Container Registry.
   - Deploys the Sitecore upgrade job.

   The databases upgrade process starts immediately.
3. Wait until the status of the job is *Complete/OK*.
4. To check the status of the job with details, run the following command:

```
kubectl get pods –o wide
```

2.4 Sitecore SQL Upgrade process

The Sitecore SQL Upgrade process:

Upgrades the databases

The database upgrade scripts contain the changes that must be applied to the Sitecore databases to ensure compatibility. They also modify certain tables and collections to support new functionality and improve performance. The changes must be applied once per database.

The Sitecore Upgrade container has all the required upgrade scripts for all the databases that must be upgraded and executes them in a specific sequence.

Cleans up the resource items in the databases

Note
If you are upgrading from Sitecore XP 10.1.X, you do not have to clean up the resource items in the database.

As the Sitecore platform evolves, some items in databases are changed or deleted, and some new items are added. In Sitecore XP 10.1.0, we have changed to an *Items as resources* strategy and the resource items have been deleted from the `Sitecore.Master` and `Sitecore.Core` databases. This enable us to keep the data in the databases consistent with the Sitecore XP version that you are upgrading to.

The Sitecore upgrade container includes a clean-up tool that automatically removes the Sitecore resource items from the databases as part of the upgrade.

Important
If you have modified the Sitecore resource items, they are not removed from the database. You must review these items individually and decide if you want to keep your modified versions or you would prefer to remove them from the database and use the new versions that are provided as resource files. We recommend that you use the new versions.

2.4.1 The database upgrade process

In Sitecore XP 10.2.0, the database upgrade process has become more flexible – you can now specify the Sitecore XP version that you are upgrading from and the version that you are upgrading to.

You specify this information in the following Docker `env` variables:

- `DATABASE_UPGRADE_FROM_VERSION`
- `DATABASE_UPGRADE_TO_VERSION`

If you do not specify these values for the database upgrade, the default values are used.

The default values are 10.1.0 and 10.2.0 respectively.

Important
The upgrade process does not count update versions. For example, upgrading the databases from 10.1.1 to 10.2.1 is the same as upgrading from 10.1.0 to 10.2.0.

Sitecore currently supports database upgrades from the Sitecore XP 9.3.0 - 10.1.X to Sitecore XP 10.2.X.

Sitecore version	XM1	XP1
9.3.0-10.2.X	List common env variables*	
		IS_ALWAYS_ENCRYPTED
		PROCESSING_ENGINE_TASKS_DATABASE_USERNAME
10.0.X-10.2.X	List common env variables*	
		IS_ALWAYS_ENCRYPTED
		PROCESSING_ENGINE_TASKS_DATABASE_USERNAME
10.1.X-10.2.X	List common env variables*	

*Use the following list of common env variables for the XM1 and XP1 topologies:

- SQL_SERVER
- SQL_USERNAME
- SQL_PASSWORD
- DATABASE_UPGRADE_FROM_VERSION
- DATABASE_UPGRADE_TO_VERSION
- SITECORE_LICENSE

For more information about the Docker `env` variables, see *Appendix C – The environment variables list.*

2.4.2 The upgrade completed criteria

The database upgrade process is carefully logged and you can find all the details of the process in the Docker container or in the Kubernetes pod logs.

The log is saved to the `C:\logs\DatabaseUpgradeScriptsLog.txt` file in the container.

When the databases upgrade process is successfully completed, the following messages appears with ExitCode 0 in the log and in the terminal:

```
INFO: Executing 'Sitecore.UpdateApp.exe' application...
Clean up resource items:
master: 7 item(s)
web: 2 item(s)
core: 0 item(s)
Resource items are cleaned up.
INFO: Database upgrade from '9.3' to '10.2' version is completed.
```

2.5 Troubleshooting

If you see the following errors in the terminal and log file:

```
INFO: Upgrading databases...
INFO: Executing 'CMS_core.sql' file...
ERROR: A network-related or instance-specific error occurred while
establishing a connection to SQL Server. The server was not found or was not
accessible. Verify that the instance name is correct and that SQL Server is
configured to allow remote connections. (provider: Named Pipes Provider,
error: 40 - Could not open a connection to SQL Server)
```

The named database cannot be accessed with the connection string provided. In this example, the problem is getting access to the *Sitecore.Core* database.

Make sure that SQL Server is accessible from the Sitecore upgrade container at the address specified in the `sql-server.txt` secret file and that the connection string for the database is correct and run the upgrade container again.

2.5.1 Build and deploy Sitecore role instances

To upgrade individual Sitecore roles in a containerized deployment, you must:

- Use the new Sitecore XP version as a codebase.
- Consume all the breaking changes and resolve all the conflicts.

  Your solution should be fully aligned with the new Sitecore version.
- Use the new Sitecore Docker images as a base in the Docker files for the corresponding roles and rebuild your images.
- Deploy the containerized environment using the new Sitecore roles images with your solution and configure it to use the upgraded SQL databases.

If you have on-premise environment, follow the instructions in the Sitecore Experience Platform Upgrade guide to upgrade your Sitecore roles. There is no need to perform any database upgrade actions as they all are done by the Sitecore upgrade container.

2.6 Clean up

When the upgrade process is completed you can clean up your environment.

Docker Compose

You must delete the upgrade container in Docker Compose.

If you ran the upgrade container in Docker Compose, from the Docker Compose folder for the topology that you upgraded, for example, sitecore-xp1, run following PowerShell cmdlet:

```
.\docker-compose.exe -f .\docker-compose.upgrade.yml --env-file .\upgrade.env down
```

This deletes the Sitecore upgrade container from the Docker system.

Kubernetes

You must delete the Kubernetes upgrade job.

If you used the Kubernetes cluster to deploy the Sitecore upgrade job, you should:

1. Start a PowerShell session from the Kubernetes upgrade artifacts folder for the target topology .
2. To delete the mssql upgrade job, run the following cmdlet:

   ```
   kubectl delete -f .\
   ```

3. To delete the upgrade secrets, run the following cmdlet:

   ```
   kubectl delete -k .\
   ```

2.7 Modules upgrade

To upgrade a Sitecore solution that has some modules installed, you must build custom mssql-upgrade image.

To build custom mssql-upgrade image, you must download the latest Sitecore XP, XM mssql-upgrade image from the container registry and create a custom Docker file on top of it.

The Sitecore XP, XM mssql-upgrade image contains configuration presets for each of the available upgrades paths.

If a module requires you to run SQL scripts, you must modify the corresponding configuration file.

The name of the configuration files has the following structure:

- `<DatabaseUpgradeFromVersion>-<DatabaseUpgradeToVersion>.json`

There are three configuration files:

- `9.3.X-10.2.X.json`
- `10.0.X-10.2.X.json`
- `10.1.X-10.2.X.json`

All the configuration files have the same structure:

```
[
    {
        "database": "<Sitecore_database_name>",
        "scripts": [
            {
                "script": "<script_name_1>.sql"
            },
            {
                "script": "<script_name_2>.sql"
            },
     …
            {
                "script": "<script_name_n>.sql"
            }
        ]
    },
    {
       "applications": [
         {
             "name": "Sitecore.UpdateApp.exe",
             "parameter": "clean"
         }
        ]
    }
]
```

If a module requires you to execute some custom SQL scripts, you must store the scripts in the `C:\data\DatabaseUpgradeScripts\<DatabaseUpgradeToVersion>\<custom SQL scripts>` folder.

To enable the mssql-upgrade container to run the script:

1. Create a powershell script that modifies the `C:\data <DatabaseUpgradeFromVersion>-<DatabaseUpgradeToVersion>.json` file.

   The script must add your instruction to the `<DatabaseUpgradeFromVersion>-<DatabaseUpgradeToVersion>.json` file which is going to run sql statements to modify the databases.

2. Save the configuration with your modifications.

3. In the Dockerfile, add a URL command that downloads the Items as Resources packages for the modules.
4. Extract the *Items as Resources* files from the downloaded module to a temporary folder

   The package for each module contains folders for the different versions of the module.
5. Select the Sitecore version from which you are going to upgrade and copy the content of the *Data* folder for the module version you are upgrading from into the `C:\data\ResourceItems\<DatabaseUpgradeFromVersion>\modules\` folder.
6. Build a custom mssql-upgrade image.

You can now run the mssql-upgrade container and use the custom upgrade image that you have just built.

To run the mssql-upgrade container with a custom image for Docker compose:

1. Update all the required parameters in the `upgrade.env` environment variable file.

   Modify the `docker-compose.upgrade.yml` file to use the image you built earlier and update the value of the image node with your image:

   ```
   image: <custome mssql-upgrade image>
   ```
2. To run the mssql-upgrade container, use the following Docker Compose command:

   ```
   docker-compose.exe -f .\docker-compose.upgrade.yml --env-file .\upgrade.env up
   ```

To run the mssql-upgrade container with a custom image for Kubernetes:

1. Update the secrets `.txt` files in the `./configuration` folder.
2. Modify the K8s specification files to use the image you built earlier and update the `mssql-upgrade.yaml`, `kustomization.yaml` file.
3. From the root folder of the desired topology, run the following command:

   ```
   kubectl apply -k .\
   ```

2.8 Upgrade Sitecore Identity Server

Note
Sitecore Identity Server 6.0.0 was released with Sitecore XP 10.2.0.

Microsoft have deprecated .NET Core runtime 2.1 and will no longer release security updates for it. We therefore strongly recommend that you use Sitecore Identity Server 6.0.0 that is based on .NET Core 3.1.

The name of the Sitecore Identity Server 6.0.0 container has been changed from *sitecore-id* to *sitecore-id6*.

When you upgrade to Sitecore 10.2.0, from a containerized Sitecore XP solution – Sitecore XP 9.3, 10.0.x, or 10.1.x – the upgrade container automatically upgrades your solution to Sitecore Identity Server 6.0.0.

Chapter 3

Appendix A – 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
```

Chapter 4

Appendix B – The Kubernetes secrets list

Name	Description	Topology	Default value
sql-server.txt	The SQL Server where the Sitecore databases that you want to upgrade are deployed.	XM1, XP1	mssql
sql-user-name.txt	SQL Server administrator username. This user should have permission to update databases schemas and execute SQL scripts against them.	XM1, XP1	
sql-password.txt	SQL Server user password that is used to execute the sql upgrade scripts and update items.	XM1, XP1	
sql-database-prefix.txt	The prefix that is used to resolve the Sitecore database names.	XM1, XP1	Sitecore
is-always-encrypted.txt	Whether *Always Encrypted* is configured for the databases to upgrade.	XP1	false
processing-engine-tasks-database-user-name.txt	The name of the least privileged user that exists in the processing_engine_tasks database.	XP1	
database-upgrade-from-version.txt	The Sitecore version from which you are going to upgrade the database	XM1, XP1	10.1.0

Name	Description	Topology	Default value
database-upgrade-to-version.txt	Sitecore version to which you are going to upgrade the database	XM1, XP1	10.2.0
sitecore-license.txt	The license file content converted to GZIP compressed and Base64 encoded string (See Appendix A)	XM1, XP1	

Chapter 5

Appendix C – The environment variables list

Name	Description	Default value
COMPOSE_PROJECT_NAME	The name of the topology.	sitecore-xm(p)1
SITECORE_DOCKER_REGISTRY	The target registry.	scr.sitecore.com/sxp/
SITECORE_VERSION	The Sitecore version that you want to upgrade to.	10.2.X-[your Windows version] For example 10.2.X-ltsc2019 or 10.2.X-2004
SQL_DATABASE_PREFIX	The prefix that is used to resolve the Sitecore database names.	Sitecore
SQL_SERVER	The SQL Server name. For more information about how to specify the connection strings for SQL Server, see Troubleshoot connecting to the SQL Server Database Engine	
SQL_USERNAME	The SQL Server administrator username.	
SQL_PASSWORD	The SQL administrator user password.	
IS_ALWAYS_ENCRYPTED	Whether *Always Encrypted* is configured for the databeses to upgrade.	

Name	Description	Default value
PROCESSING_ENGINE_TASKS_DATABASE_USERNAME	The name of the least privileged user that exists in the processing_engine_tasks database.	
DATABASE_UPGRADE_FROM_VERSION	The Sitecore version from which you are going to upgrade the database	10.1.0
DATABASE_UPGRADE_TO_VERSION	The Sitecore version to which you are going to upgrade the database	10.2.0
SITECORE_LICENSE	The Sitecore license file converted to base 64 string through PowerShell script.	
ISOLATION	Override for Docker isolation level Possible values: default, hyperv, process	default