SITECORE

Installation Guide for a Developer Workstation with Containers

Sitecore Experience Commerce 10.0.0

October 14, 2021

Table of Contents

1. Introduction

Sitecore Experience Commerce 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 Commerce.

This guide provides step-by-step instructions for installing Sitecore Experience Commerce on a developer workstation, based on the sample images that are included in the Sitecore Commerce Container SDK package, available on the Sitecore Downloads site.

For information on installing Sitecore Experience Commerce using Kubernetes in a production environment, download the Sitecore XC Installation Guide for Production Deployments with Kubernetes.

1.1. Topologies

The Sitecore Commerce Container SDK provides container images that allows you to install Sitecore XC on a developer workstation in the following topologies:

- XC Workstation topology (XC0)
- XC Scaled topology (XC1 and SX1-CXA)

1.1.1. XC Workstation topology (XC0)

The XC0 topology is used to deploy the Commerce Engine only, in a Docker workstation developer environment. This topology is not intended for production deployment. It is designed to reduce memory overhead, reduce download size, improve startup/shutdown time, and reduce complexity.

> **NOTE**
> The XC0 topology for developer workstation *does not* include container images for the SXA storefront or supporting modules.
>
> For a deployment that includes the modules required to deploy the SXA storefront, you must install the XC1-CXA topology.

1.1.2. XC Scaled topologies (XC1 and XC1-CXA)

The Sitecore Commerce Container SDK includes two XC scaled topologies:

- The XC1-CXA topology is for a scaled Commerce deployment. The XC1-CXA topology includes container images of modules required to support the SXA Storefront. This is the topology you need to deploy if you want the SXA Storefront in your development environment.

  > **NOTE**
  > The XC1-CXA topology does not include a default storefront site configuration. The topology provides container images of all supporting modules required to create a tenant and SXA Storefront site as post-deployment steps.

- The XC1 topology is for a scaled deployment that *excludes* modules supporting the SXA Storefront. You use this topology in a scaled, Commerce Engine only deployment, that does not use the SXA Storefront.

  > **NOTE**
  > The Sitecore Commerce Container SDK does not include all Docker compose YAML files required to deploy the container images of the XC1 topology.

The resources required to run XC in a non-production deployment can be significant but are required to mimic the exact configuration that is used in production. In non-production deployments, we recommend that you run XC with the workstation hardware requirements.

1.1.3. Roles included in XC0 and XC1-CXA topologies

The XC0 and XC1-CXA topologies include container images for Sitecore roles and for external services.

The following table lists the Sitecore roles that are included in the XC0 and XC1-CXA topologies, respectively.

Sitecore roles	XC0 topology	XC1-CXA topology
Sitecore Identity Server	✓	✓
Sitecore Commerce Business Tools (BizFX)	✓	✓
Sitecore Commerce Authoring	✓	✓
Sitecore Commerce Shops	✓	✓
Sitecore Commerce Minions	✓	✓
Sitecore Commerce Ops	✓	✓
XDB Search Worker	✓	✓
Marketing Automation Engine	✓	✓
Sitecore Cortex Processing Engine	✓	✓
xConnect Server (Standalone)	✓	
Content Management (Standalone CM/CD)	✓	
xConnect Server		✓
Content Management		✓
Content Delivery		✓
XDB Processing		✓
XDB Reporting service		✓
XDB Collection service		✓
XDB Search service		✓
XDB Reference Data service		✓
Marketing Automation Reporting		✓

In addition to the Sitecore roles, the XC0 and XC1-CXA topologies include sample container images of external services.

NOTE
The external services are provided as non-production container images to facilitate the deployment of the Sitecore XC sample solution in a development environment. They are not intended nor suitable for production deployment.

- Microsoft SQL Server
- Apache Solr
- RedisLabs Redis Server
- Traefik Reverse Proxy

1.2. About production and non-production containers images

Sitecore XC Docker images that are provided as samples are not suitable for a production environment.

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.

IMPORTANT

The non-production services do not follow the best practices for hosting a production environment and should not be considered as a basis for production environments.

1.3. Sitecore XC Docker Compose files

Overview of how Docker Compose files are used in a Sitecore XC deployment and the information they contain about containers and configurations.

To deploy the containers, Sitecore Docker Compose requires the following files:

- The `docker-compose.yml` file
- The `.env` file

NOTE
The Docker Compose files for each Sitecore XC topology are included in the Sitecore Commerce Container SDK.

1.3.1. Docker Compose file

The Docker Compose configuration file is a text file (`docker-compose.yml`) that contains information about the different containers and configuration of each Sitecore role in the deployment topology. The Docker Compose file is the main configuration file that the `docker-compose` command uses.

1.3.2. Environment variables file

The environment variable configuration file is a text file (`.env`) that stores the configuration information for the environment you want to deploy. The sample `.env` file provides default values for any environment variables referenced in the Docker Compose file. You can edit this file outside the main Docker Compose configuration, that is, using a text editor without using docker-compose command line utility.

The environment variables are the preferred mechanism for passing configuration settings into containers.

The following example shows how, for example, the `mssql` service role in the `compose.yml` file uses an environment variable defined in the `.env` file to configure the SQL Server SA password (`SA_PASSWORD`):

~~~
``` yaml
    mssql:
        isolation: ${ISOLATION}
        image: ${XC_NONPRODUCTION_SITECORE_DOCKER_REGISTRY}sitecore-xc0-mssql:${XC_PACKAGES_TAG}
        environment:
          SA_PASSWORD: ${SQL_SA_PASSWORD}
          SITECORE_ADMIN_PASSWORD: ${SITECORE_ADMIN_PASSWORD}
          ACCEPT_EULA: "Y"
          SQL_SERVER: mssql
        ports:
          - "14330:1433"
        volumes:
          - type: bind
            source: c:\containers\mssql-data
            target: c:\data
```
~~~

1.4. Software Requirements

The following are software requirements for installing Sitecore Experience Commerce with containers on your developer workstation:

- Operating system:
  - Windows 10 - The most recent 2 semi-annual feature releases.
    or
  - Windows Server - The current LTS version and most recent 2 semi-annual feature releases.
- Docker Desktop for Windows
- Windows PowerShell 5.0
- Sitecore Commerce Container SDK

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

See this article for additional details about Sitecore XC software compatibility for this release.

1.5. Hardware requirements

The following are hardware requirements for installing Sitecore Experience Commerce with containers on your developer workstation:

- RAM
  We recommend that a developer workstation has a minimum of 32GB of RAM.
- CPU
  We recommend a quad core or higher.
- Disk
  Sitecore container images require approximately 25 GB of free disk space. We recommend the use of solid-state drive (SSD) disks for optimal performance when downloading and running Docker containers.

NOTE
The type of disks used for SQL Server and Solr can have a significant impact on performance.

2. Prepare for Commerce containers deployment

Before you can deploy the Sitecore XC containers, you must perform the following tasks to prepare required files and certificates:

- Download the `Sitecore.Commerce.Container.SDK.*.*.*.ZIP` package and familiarize yourself with its content. For a description of the SDK, see this topic .
- Prepare the environment variables
- Generate the Identity Server token signing certificate
- Generate TLS/HTTPS certificates
- Update the Windows hosts file
- Read about production and non-production container images

2.1. Prepare the environment variables file

Environment variables are the preferred mechanism for passing configuration settings into Sitecore containers.

The environment variables for Sitecore Docker Compose are stored in an environment variable configuration file (`.env`). Docker Compose loads these variables automatically during startup.

IMPORTANT

All the environment variables must fit inside a single 32,700 character block in the `.env` file. If the total size of your own variables combined with the Sitecore variables exceeds this size, you will be unable to set new values, and the system will not deploy successfully.

NOTE

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

You use the `UpdateEnvCompose.ps1` script to prepare the `.env` file. The script sets values for those variables who do not have a default value defined in the `.env` file. For a brief description of all variables contained in the `.env` file, see the environment variables list.

Following is the list of variables for which you must provide a value, and that are propagated to the `.env` file when you run the `UpdateEnvCompose.ps1`:

Script parameter	Description
`licenseFile`	The path to your Sitecore license file. Example: "`C:\Docker\Sitecore.Commerce.Container.SDK.1.0.214\scripts\license.xmlC:\temp\license.xml` In the env. file, sets the `SITECORE_LICENSE` variable.
`braintreeEnvironment`	The Braintree environment where to direct API requests. Example: "`sandbox`" In the `env.` file, sets the `XC_ENGINE_BRAINTREEENVIRONMENT` variable.
`braintreeMerchantId`	Your merchant ID for the Braintree payment provider. Example: "`rwd84b5k2rck8c7f`" In the `env.` file, sets the `XC_ENGINE_BRAINTREEMERCHANTID` variable.
`braintreePublicKey`	The public key associated to your Braintree account. Example: "`747f5tsgkbk9xrk3`" In the variable `env.` file, sets the `XC_ENGINE_BRAINTREEPUBLICKEY` variable.
`braintreePrivateKey`	The private key associated to your Braintree account. Example: "`25d3cfa5a9674a8b12e167af80b8607f`" In the variable `env.` file, sets the `XC_ENGINE_BRAINTREEPRIVATEKEY` variable.

Script parameter	Description
`telerikKey`	The symmetric key used by the Telerik web controls. Length: 64-128 characters random string. Example: "`mPgyfFKEf70UmVoE74LY93RmieAujDOD`" In the `env.` file, sets the `TELERIK_ENCRYPTION_KEY` variable.
`idCert`	The ID Service certificate used to encrypt data. Example: "`mPgyfFKEf70UmVoE74LY93RmieAujDOD` In the `env.` file, sets the `SITECORE_ID_CERTIFICATE` variable.
`idPassword`	The password used in the script that creates the ID Service certificate, and that is required to open the Identity Server certificate. Example: "`Test123!`" In the `env.` file, sets the `SITECORE_ID_CERTIFICATE_PASSWORD` variable.
`idSecret`	The shared secret between the Identity Server and client roles. Length: 64 characters Example: "`utxHufWfiDEuqCK9a1kQ2sBvbX83gHMpVsrqptkvoOMttDjsvqrMmHwRG33aBYgL`" In the `env.` file, sets the `SITECORE_IDSECRET` variable.
`xcIdSecret`	The shared secret to use as a salt when generating hash values between the Identity Server and Commerce Engine Connect client roles. Length: 64 characters. Example: "`utxHufWfiDEuqCK9a1kQ2sBvbX83gHMpVsrqptkvoOMttDjsvqrMmHwRG33aBYgL`" In the `env.` file, sets the `XC_IDENTITY_COMMERCEENGINECONNECTCLIENT_CLIENTSECRET1` variable.
`reportingApiKey`	The symmetric key - ASCII string can be any combination of numerals or text - used to access the Sitecore XDB Reporting WebAPI. Length: 32 characters Example: "`GvkOg8s4jOgGN0SzBOq4J8rDwXyOZKR8`" In the `env.` file, sets the `REPORTING_API_KEY` variable.

To prepare the environment variable:

- Open a PowerShell window, replace the values with your own values, and run the following sample script:

```
/UpdateEnvCompose.ps1 -envRootPath 'C:\Docker\Sitecore.Commerce.Container.SDK.*.*.***\xc0' -
licenseFile "C:\Docker\Sitecore.Commerce.Container.SDK.*.*.***\scripts\license.xml" -
braintreeEnvironment "sandbox" -braintreeMerchantId "rwd84b5k2rck8c7f"
 -braintreePublicKey "747f5tsgkbk9xrk3" -braintreePrivateKey
"25d3cfa5a9674a8b12e167af80b8607f" -telerikKey "mPgyfFKEf70UmVoE74LY93RmieAujDOD" -idCert
"MIIKqQIBAzCCCmUGCSqGSIb3DQEHAaCCC…" -idSecret
"utxHufWfiDEuqCK9a1kQ2sBvbX83gHMpVsrqptkvoOMttDjsvqrMmHwRG33aBYgL"
 -idPassword "Test123!" -xcIdSecret
"srNMmM8lPeAYCcABY90f6nJIWWgx1DVX1ldc2iNN0z3qVZcGzuKwfGEHKDJzNMZ8" -reportingApiKey
"GvkOg8s4jOgGN0SzBOq4J8rDwXyOZKR8" -mediaSecret "25d3cfabDRS674a8r93Ab12e167a2Jk07f"
```

NOTE

The script creates the following Docker container volumes under `"c:\containers"`:

- `"cm\domains-shared"`
- `"cd\domains-shared"`
- `"engine\catalogs"`
- `"mssql-data"`
- `"solr-data"`

2.1.1. The environment variables list

The following table lists the environment variables contained in the `.env` file for each Sitecore topology.

NOTE

The variables that do not have a default value defined are those that are set when you run the `UpdateEnvCompose.ps1` script.

Environment variables for XC0, XC1, and XC1-CXA topologies

Variable name	Topology	Default value	Description
`BASE_SITECORE_DOCKER_REGISTRY`	XC0, XC1, XC1-CXA	`src.sitecore.com/base/`	The base Sitecore container registry.
`XP_SITECORE_DOCKER_REGISTRY`	XC0, XC1, XC1-CXA	`scr.sitecore.com/sxp/`	Sitecore container registry.
`XP_SITECORE_TAG`	XC0, XC1, XC1-CXA	`10.0.0-ltsc2019` **NOTE** Refer to the Sitecore.Commerce.Container.SDK for the latest value.	Image tag with the version to be pulled from the container registry.
`MODULES_SITECORE_DOCKER_REGISTRY`	XC1, XC1-CXA	`scr.sitecore.com/sxp/modules/`	The Sitecore modules container registry.
`SPE_SITECORE_TAG`	XC1-CXA	`6.1.1-1809` **NOTE** Refer to the Sitecore.Commerce.Container.SDK for the latest value.	The container image tag for the Sitecore Powershell Extension module.

Variable name	Topology	Default value	Description
`SXA_SITECORE_TAG`	XC1-CXA	`10.X.X-1809` **NOTE** Refer to the Sitecore.Commerce.Container.SDK for the latest value.	The container image tag for the Sitecore SXA module.
`XC_NONPRODUCTION_SITECORE_DOCKER_REGISTRY`	XC0, XC1-CXA	`scr.sitecore.com/sxc/nonproduction`	The Sitecore Commerce container registry for non-production container.
`XC_SITECORE_DOCKER_REGISTRY`	XC0, XC1-CXA	`scr.sitecore.com/sxc/`	Sitecore Commerce container registry.
`XC_PACKAGES_TAG`	XC0, XC1, XC1-CXA	`10.0.0-ltsc2019` **NOTE** Refer to the Sitecore.Commerce.Container.SDK for the latest value.	Image tag with the version to be pulled from the container registry.
`TRAEFIK_IMAGE`	XC0, XC1, XC1-CXA	`traefik:v2.2.0-windowsservercore-1809`	The Traefik image tag.
`TRAEFIK_ISOLATION`	XC0, XC1, XC1-CXA	`default`	Override for Docker isolation level for traefik. Possible values: default, hyperv, process.
`ISOLATION`	XC0, XC1-CXA	`default`	Override for Docker isolation modes. Possible values: default, hyperv, process.
`CD_HOST`	XC1-CXA	`xc1cd.localhost`	CD host name.
`CM_HOST`	XC0, XC1-CXA	For XC0: `xc0cm.localhost` For XC1-CXA: `xc1cm.localhost`	CM host name.
`ID_HOST`	XC0, XC1-CXA	For XC0: `xc0id.localhost` For XC1-CXA: `xc1id.localhost`	ID Server host name.
`AUTHORING_HOST`	XC0, XC1-CXA	`authoring.localhost`	Authoring service host name.
`SHOPS_HOST`	XC0, XC1-CXA	`shops.localhost`	Shops service host name.
`MINIONS_HOST`	XC0, XC1-CXA	`minions.localhost`	Minions service host name.
`OPS_HOST`	XC0, XC1-CXA	`ops.localhost`	The Ops service host name.

Variable name	Topology	Default value	Description
`BIZFX_HOST`	XC0, XC1-CXA	`bizfx.localhost`	The Business Tools host name.
`SITECORE_ADMIN_PASSWORD`	XC0, XC1-CXA	`Password12345`	The Sitecore application administrator password.
`SQL_SA_PASSWORD`	XC0, XC1-CXA	`Password12345`	The SQL Server administrator password.
`SITECORE_MASTER_DB`	XC0, XC1, XC1-CXA	`Sitecore.Master`	The Sitecore master db name.
`SITECORE_CORE_DB`	XC0, XC1, XC1-CXA	`Sitecore.Core`	The Sitecore core db name.
`XC_GLOBAL_DB`	XC0, XC1, XC1-CXA	`SitecoreCommerce_Global`	The Sitecore Commerce global db name.
`XC_GLOBAL_DB_TRUSTED_CONNECTION`	XC0, XC1-CXA	`false`	Whether the connection is trusted or not.
`XC_SHARED_DB`	XC0, XC1 XC1-CXA	`SitecoreCommerce_SharedEnvironments`	The Sitecore Commerce shared db name.
`XC_SHARED_DB_TRUSTED_CONNECTION`	XC0, XC1-CXA	`false`	Whether the connection is trusted or not.
`XC_ENGINE_BRAINTREEENVIRONMENT`	XC0, XC1-CXA	*no default*	The Braintree environment. Set by running the `UpdateEnvCompose.ps1` script.
`XC_ENGINE_BRAINTREEMERCHANTID`	XC0, XC1-CXA	*no default*	Your merchant ID for the Braintree payment provider. Set by running the `UpdateEnvCompose.ps1` script.
`XC_ENGINE_BRAINTREEPUBLICKEY`	XC0, XC1-CXA	*no default*	The public key associated to your Braintree account. Set by running the `UpdateEnvCompose.ps1`script.
`XC_ENGINE_BRAINTREEPRIVATEKEY`	XC0, XC1-CXA	*no default*	The private key associated to your Braintree account. Set by the running the `UpdateEnvCompose.ps1` script.
`XC_BIZFX_DEFAULT_LANGUAGE`	XC0, XC1-CXA	`en`	The default language used in the Business Tools user interface.
`XC_BIZFX_DEFAULT_CURRENCY`	XC0, XC1-CXA	`USD`	The default currency to use by the shop.
`XC_BIZFX_DEFAULT_SHOPNAME`	XC0, XC1-CXA	`CommerceEngineDefaultStorefront`	The default shop name.

Variable name	Topology	Default value	Description
`XC_ENGINE_CONNECT_CLIENTID`	XC0, XC1-CXA	`CommerceEngineConnect`	The client ID assigned to Commerce Engine Connect for Sitecore Identity. This ID is used to identify the Commerce Engine Connect with Commerce Engine.
`XC_IDENTITY_ COMMERCEENGINECONNECTCLIENT_ CLIENTSECRET1`	XC0, XC1-CXA	*no default*	Shared secret between the Identity Server and Commerce Engine Connect client roles. Length: 64 characters Set by the running the `UpdateEnvCompose.ps1` script.
`REPORTING_API_KEY`	XC0, XC1-CXA	*no default*	Symmetric key used to access the Sitecore XDB Reporting WebAPI. Length: 32 characters Set by the running the `UpdateEnvCompose.ps1` script.
`TELERIK_ENCRYPTION_KEY`	XC0, XC1-CXA	*no default*	Symmetric key used by the Telerik web controls. Length: 64-128 characters Set by the running the `UpdateEnvCompose.ps1` script.
`SITECORE_IDSECRET`	XC0, XC1-CXA	*no default*	Shared secret between the Identity Server and client roles. Length: 64 characters Set by running the `UpdateEnvCompose.ps1` script.
`SITECORE_ID_CERTIFICATE`	XC0, XC1-CXA	*no default*	The Identity Server certificate used to encrypt data. Set by the running the `UpdateEnvCompose.ps1` script.
`SITECORE_ID_CERTIFICATE_PASSWO RD`	XC0, XC1-CXA	*no default*	The password required to open the Identity Server certificate. Set by the running the `UpdateEnvCompose.ps1` script.
`SITECORE_LICENSE`	XC0, XC1-CXA	*no default*	The path to the Sitecore license file.

2.2. Generate 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. You must generate this certificate, and encode it to a Base64 encoded string form.

To generate a self-signed certificate :

NOTE
The following shows 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.

- Run the following sample script:

  NOTE
  The value of the password to convert (in the following example `"Test123!"`) must match the value of the `"Sitecore_ID_Certificate_Password"` variable specified in the `.env` file, or the `"idPassword"` value used in the `UpdateEnvCompose.ps1` file.

```
$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
```

  In the `env.` file, the output of this script provides the values for the `SITECORE_ID_CERTIFICATE` environment variable.

  NOTE
  Alternatively, like the Sitecore license file, you can also mount the Sitecore Identity Server certificate on the file system instead of passing it as an environment variable.

2.3. Generate 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.

> **NOTE**
> The Sitecore Commerce Container SDK contains dummy certificates you can use to get started in the folder `/xc0/traefik/`.
>
> If you change the local host names (in case, for example, that you have multiple Commerce instances running in parallel), you must generate your own certificates.

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.

All communication between the Traefik edge router and the individual containers is encrypted with the HTTPS protocol.

HTTPS protocol is required to support the secure browser cookies used by the Sitecore Content Management role and the Identity Server role.

To generate TLS/SSL certificates:

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

   > **NOTE**
   > The `mkcert` utility will prompt the user the first time to install the generated self-signed root certificate authority.

   Sample script for an XC0 topology:

   ```
   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\xc0cm.localhost.crt -key-file traefik\certs
   \xc0cm.localhost.key "xc0cm.localhost"

   mkcert -cert-file traefik\certs\xc0id.localhost.crt -key-file traefik\certs
   \xc0id.localhost.key "xc0id.localhost"

   mkcert -cert-file traefik\certs\authoring.localhost.crt -key-file traefik\certs
   \authoring.localhost.key "authoring.localhost"

   mkcert -cert-file traefik\certs\shops.localhost.crt -key-file traefik\certs
   \shops.localhost.key "shops.localhost"
   ```

```
mkcert -cert-file traefik\certs\minions.localhost.crt -key-file traefik\certs
\minions.localhost.key "minions.localhost"

mkcert -cert-file traefik\certs\ops.localhost.crt -key-file traefik\certs
\ops.localhost.key "ops.localhost"

mkcert -cert-file traefik\certs\bizfx.localhost.crt -key-file traefik\certs
\bizfx.localhost.key "bizfx.localhost"
```

Example for a XC1-CXA topology:

```
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\xc1cm.localhost.crt -key-file traefik\certs
\xc1cm.localhost.key "xc1cm.localhost"

mkcert -cert-file traefik\certs\xc1cd.localhost.crt -key-file traefik\certs
\xc1cd.localhost.key "xc1cd.localhost"

mkcert -cert-file traefik\certs\xc1id.localhost.crt -key-file traefik\certs
\xc1id.localhost.key "xc1id.localhost"

mkcert -cert-file traefik\certs\authoring.localhost.crt -key-file traefik\certs
\authoring.localhost.key "authoring.localhost"

mkcert -cert-file traefik\certs\shops.localhost.crt -key-file traefik\certs
\shops.localhost.key "shops.localhost"

mkcert -cert-file traefik\certs\minions.localhost.crt -key-file traefik\certs
\minions.localhost.key "minions.localhost"

mkcert -cert-file traefik\certs\ops.localhost.crt -key-file traefik\certs
\ops.localhost.key "ops.localhost"

mkcert -cert-file traefik\certs\bizfx.localhost.crt -key-file traefik\certs
\bizfx.localhost.key "bizfx.localhost"
```

2.4. Update the Windows hosts file

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

The following shows an example of the content of Windows host file definition with the default hostnames for the XC0 topology:

```
127.0.0.1 xc0cm.localhost
127.0.0.1 xc0id.localhost
127.0.0.1 bizfx.localhost
127.0.0.1 authoring.localhost
127.0.0.1 shops.localhost
127.0.0.1 ops.localhost
```

The following table lists the default hostnames for each topology:

Topology	IP address	Hostnames
XC Workstation (XC0)	127.0.0.1	xc0cm.localhost
	127.0.0.1	xc0id.localhost
	127.0.0.1	bizfx.localhost
	127.0.0.1	authoring.localhost
	127.0.0.1	shops.localhost
	127.0.0.1	ops.localhost
XC Scaled (XC1)	127.0.0.1	xc1cm.localhost
	127.0.0.1	xc1cd.localhost
	127.0.0.1	bizfx.localhost
	127.0.0.1	authoring.localhost
	127.0.0.1	shops.localhost
	127.0.0.1	minions.localhost
	127.0.0.1	ops.localhost

To change the default host names, you must:

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

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

3. Deploy a Sitecore XC developer workstation

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

To deploy Sitecore XC developer workstation using containers:

1. In Docker for Windows, switch to Windows container mode. (The Docker for Windows tray icon has an option to switch between Linux and Windows container development).
2. Download and extract the `Sitecore.Commerce.Container.SDK.*.*.zip` package from the Sitecore Developer Portal and store it on your local workstation.
3. In Windows Explorer, go to the folder where you extracted the Sitecore Commerce Container SDK, 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 license file.
5. If required, to generate the required TLS reverse proxy certificates, open a command prompt/ terminal, and run the sample shell script.
6. If you did not generate your own self-signed root certificate, you can install the packaged dummy certificate. To install the packaged dummy root certificate, open the `./traefik/certs` folder, double- click the `root-ca.crt` file, and follow the prompts.
7. Open the `/xc0/docker-compose.yml` file, and review the content to get a better understanding of the containers and connection strings between the different roles.
8. In the Windows console, go to the folder that contains the `docker-compose.yml` file, and run the following Docker Compose command:

   ```
   docker-compose up --detach
   ```

   NOTE
   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.

   IMPORTANT
   Before running an image, ensure that your host operating system version is greater than or equal to the OS version of the container image. Otherwise, the following error may occur: "ERROR: manifest not found: manifest unknown: manifest unknown" error."

   Also, ensure the host OS version patch level matches or is greater than the container OS image patch level.

9. To check the Docker container status, run the following command:

   ```
   docker container list
   ```

NOTE
This command generates a list of all the containers and their current status.

10. When the status of all the containers is healthy, validate that you can access the Commerce Authoring environment. Open a browser and enter the URL for the instance of the Commerce Engine running the Commerce Authoring service. The default host name for the Commerce Authoring is: `https://authoring.localhost/commerceops/$metadata`. (The default host names for the other topologies are listed here.)

    NOTE
    Commerce services run on the following ports by default: 443, 8079 and 8080. To avoid errors, ensure to stop all services running on these ports.

11. When deployment is done, you must complete the post-deployment steps.

4. Post-deployment tasks

Once you have confirmed that the status of all containers is healthy, you must perform the following tasks to complete your deployment:

- Bootstrap and initialize the Commerce Engine
- Validate the deployment of the Business Tools
- Configure user accounts
- Generate catalog templates
- Perform full rebuild of Commerce indexes
- Populate Solr managed schema
- Manually rebuild Sitecore XP indexes
- Create an SXA Storefront tenant and site (for XC1-CXA topology only)
- Clean up a workstation environment

4.1. Bootstrap and initialize the Commerce Engine

After you have confirmed that Docker containers have a healthy status, you must bootstrap and initialize your Commerce environments.

The Sitecore Commerce Engine SDK includes samples of API calls for DevOps operations, so that you can access the Sitecore XC API directly. The following instructions assume that you are using Postman to exercise the Sitecore XC API.

NOTE
The following instructions assume that you have access to a Sitecore XC development (or DevOps) environment, with the Postman API samples deployed. The Postman samples are included as part of the Sitecore Commerce Engine SDK, available for download in *Sitecore XC Packages for On Premise WDP*.

4.1.1. Setup the environment in Postman

You must setup the environment in Postman to point to your deployment before you can exercise the API samples.

NOTE
With a new installation of Postman, you must disable SSL certificate verification in order to get a response back from the Commerce Engine. To do this, in Postman, click **File**, **Settings** and then set `SSL certificate verification` to *OFF*.

To setup the environment in Postman, for example, the *Habitat Environment*:

1. In the top right corner of Postman, click **Settings**.
2. In the **Manage Environments** dialog, click the *Habitat Environment*, and update the values to match those defined in the `.env` file from the topology folder.
   The following table shows the default values from the `Sitecore.Commerce.Container.SDK/xc0/.env`)

Variable	Current value
Environment	HabitatAuthoring
ShopperId	ShopperId
Language	en-US
Currency	USD
ServiceHost	https://authoring.localhost
OpsApiHost	https://ops.localhost
AuthoringHost	https://authoring.localhost
MinionsHost	https://minions.localhost
ShopsHost	https://shops.localhost

Variable	Current value
SitecoreIdServerHost	https://xc0id.localhost **NOTE** For XC1-CXA deployment, the URL is https://xc1id.localhost.
HostName	Not required
SitecoreIdServerPassword	The admin user password used to authenticate the request.

4.1.2. Bootstrap and initialize the Commerce Engine

To run the bootstrap and initialize operations:

1. In the top right corner of Postman, click the environment selector and select the environment, for example the *Habitat Environment*.
2. In the Postman **Collections** pane, open the *Authentication* folder, and in the *Sitecore* sub-folder, execute the `GetToken` request.
3. Open the *SitecoreCommerce_DevOps* folder.
4. Open the *1 Environment Bootstrap* folder, and execute the **Bootstrap Sitecore Commerce** call.
5. Open the *3 Environment Initialize* folder, and execute the **Ensure\Sync default content paths** call.

   NOTE
   If the `status` of a request is `WaitingForActivation`, you can execute the Check Long Running Command Status request. When you execute the `CheckCommandStatus` request, you must ensure you are calling the same service that the previous command was executed in.

6. In the *3 Environment Initialize* folder, execute the **Initialize Environment** call.

   NOTE
   If the `status` of a request is `WaitingForActivation`, you can execute the Check Long Running Command Status request. When you execute the `CheckCommandStatus` request, you must ensure you are calling the same service that the previous command was executed in.

7. Repeat step 4 and 5 above for other environments if applicable (for example, for the AdventureWorks environment).

4.2. Validate the deployment

After bootstrapping and initializing the Commerce Engine, make sure that you can access the Business Tools, and the Sitecore Launchpad.

4.2.1. Validate the deployment of Business Tools

The Sitecore Commerce XC Business Tools are deployed in the Authoring environment.

To validate the deployment of the Business Tools:

1. Open a browser, and enter the URL for the Commerce Business tools instance. The default host name for the XC Business Tools is: `https://bizfx.localhost`.
2. Login to the Business Tools and ensure that you can browse the tools.

> **NOTE**
> Within the Sitecore Launchpad, the links to the Business Tools will be broken when you bring up the containers. To fix it, follow these instructions and, in the **Link** field, enter the URL `https://bizfx.localhost/` (instead of `https://localhost:4200`).

4.2.2. Validate access to the Content Management instance

To validate the deployment of the Content Management instance:

1. Open a browser, and enter the URL for the Content Management instance. The Content Management instance runs on port 443 and uses the HTTPS protocol.
   The default host name for the Content Management instance is:
   - In a XC0 topology: `https://xc0cm.localhost`
   - In a XC1-CXA topology: `https://xc11cm.localhost`
2. Validate that you can login to Sitecore and access the Sitecore Launchpad.

4.3. Configure user accounts

After you have deployed your Sitecore XC solution, you must create user accounts and assign the appropriate roles.

NOTE
Every Sitecore XC user who requires access to the Business Tools must have the *Commerce Business User* role assigned, at a minimum.

You create users and assign roles using the **User Manager** tool on the **Sitecore Launchpad**.

Refer to the User roles and permissions topic for information on the pre-defined roles and associated permissions for the Sitecore XC Business Tools.

4.4. Generate catalog templates

After you have deployed your Sitecore XC solution, you must refresh the cache and generate catalog templates, then republish the site.

You can perform both of these operations from the **Content Editor** on the **Sitecore Launchpad**.

To generate catalog templates:

1. Open a browser, and login to the **Sitecore Launchpad** (in a container deployment, the URL is `https://cm.globalhost/sitecore`, or in an Azure deployment, the URL is `https://<deployment name>-cm.azurewebsites.net/sitecore`).
2. Click on **Content Editor**.
3. In the Content Editor, click on the **Commerce** tab.
4. Click on **Refresh Commerce Cache** (in the **Caches** tile).
5. Click on **Update Data Templates** (in the **Catalog** tile).

4.5. Perform full rebuild of Commerce indexes

After you initialized your environments with Commerce data (for example, the sample AdventureWorks or Habitat environments), you must rebuild the following Commerce indexes using Postman:

- Catalog Items
- Promotions
- Price Cards

To rebuild Commerce search indexes using Postman :

1. In the Postman **Collections** pane, expand the *SitecoreCommerce_DevOps* collection.
2. Open the *Minions* folder, and execute the following request:
   - *Run FullIndex Minion - Catalog Items* request.
   - *Run FullIndex Minion - Promotions* request.
   - *Run FullIndex Minion - PriceCards*

4.6. Create an SXA Storefront tenant and site

If your deployment topology includes the SXA Storefront and you to want to use the SXA Storefront site as a starting point to create your own e-commerce site, you must create a new tenant and storefront site using this procedure..

NOTE
In deployment using Kubernetes, the SXA Storefront site is functional only after you complete all post-deployment steps.

4.7. Clean up a workstation environment

You can stop or completely remove workstation environment.

To stop a Docker Compose environment without removing its contents, run:

```
docker-compose stop
```

To resume a previously stopped Docker Compose environment, run:

```
docker-compose start
```

To remove a Docker Compose environment and all the non-mounted volumes, run

```
docker-compose down
```