Sitecore Experience Commerce Installation Guide for the Azure App Service

Sitecore XC 9.1

May 9, 2019

Table of Contents

1. Getting started

This guide describes how to install Sitecore Experience Commerce™ (XC) in an Azure App Service configuration.

The Microsoft Azure App Service® is a cloud-computing platform that provides a rich variety of services to help you create and run scalable applications without high upfront infrastructure investments.

For details on installing Sitecore XC in a single-server on-premises configuration, see the *Sitecore Experience Commerce Installation Guide for On-Premise Solutions*.

1.1. Sitecore Experience Commerce solution

Sitecore Experience Commerce (Sitecore XC) is an e-commerce solution, built on the Sitecore Experience Platform (Sitecore XP).

The Sitecore XC solution provides a core framework for rapidly delivering commerce functionality through the following components:

- **Commerce Engine**
  An extensible commerce core framework, hosting commerce services such as Cart, Order, Pricing, Promotions, Catalogs, and Inventory. The Commerce Engine includes a pluggable framework for extending the engine to modify or add to existing functionality.
- **Commerce Business Tools**
  A set of rich business tools for merchandisers and customer service representatives. The business tools are built on the Angular framework, and can also be extended using the same pluggable framework.
- **Sitecore Experience Accelerator (SXA) Storefront**
  A sample storefront website that is integrated with the Commerce Engine. You can use the SXA Storefront as a starting point for building a customized storefront.

1.2. Overview of the installation process

You can install your Sitecore XC solution to Azure using the Sitecore Marketplace application, or by using the Sitecore Azure Toolkit.

The installation process for deploying your Sitecore XC solution to Azure using the Sitecore Azure Toolkit has several tasks.

To install an instance of Sitecore Experience Commerce to the Azure App Service:

1. Prepare your environment by completing the following prerequisite tasks:
    - Set up Azure storage.
    - Obtain a valid client certificate.
    - Create a Braintree sandbox account (for web payment functionality).
2. If you are installing your Sitecore XC solution using the Sitecore Azure Toolkit, you must do the following:
    - Install the Sitecore Azure Toolkit.
    - Obtain the Web Deploy Packages (WDPs) for Sitecore XP, Sitecore XC, and Sitecore Experience Accelerator (SXA), then upload the WDPs to Azure storage.
    - Configure the Azure Resource Manager (ARM) templates and specify the values for your environment in the ARM parameters file.
3. Deploy the Sitecore XC solution to Azure.
4. Complete the post-installation set-up steps to finish the installation.

1.3. Sitecore XC packages for Azure App Service

The Sitecore XC Azure release package contains Web Deploy Packages (WDPs) for Sitecore XC-specific components. The Azure release package does not include any Sitecore XP software.

The Sitecore XP and Sitecore Experience Accelerator (SXA) WDPs are available separately. You can download Sitecore XP WDPs from the Sitecore Experience Platform Download page.

You can download Sitecore Experience Accelerator (SXA) WDPs from the Sitecore Experience Accelerator Download page.

The following tables lists the Web Deploy Packages (WDPs) required for your Sitecore XC cloud installation. Each package has a unique version number and a `.scwdp.zip` extension.

Package	Description
Sitecore.Commerce.Engine.Azure	Contains the binary for the Commerce Engine, with support for Azure Search as search provider.
Sitecore.Commerce.Engine.Solr	Contains the binary for the Commerce Engine, with support for Solr as search provider.
SolrSchemas.Sitecore.Commerce	Contains Solr schema files for SolrCloud search configuration.
Sitecore.Identity.Config.Commerce	Contains configuration for interaction with the Sitecore Identity Server.
Sitecore.BizFx	Contains an integrated suite of Sitecore XC business tools, built on the Angular application platform version 4.
Sitecore.Commerce.Habitat.Images	Contains images for the Habitat sample catalog.
Adventure Works Images	Contains images for the Adventure Works sample catalog.
Sitecore Commerce Connect Core	Contains a middleware integration layer between the Sitecore XC back-end and the front-end Storefront, and with the Sitecore XP.
Sitecore Commerce Connect Schema Definitions for IndexWorker	Contains schema definitions for the Commerce Connect IndexWorker.
Sitecore Commerce Connect Definitions for xConnect	Contains definitions for xConnect.
Sitecore Commerce Engine Connect CD Azure	Contains Commerce Engine Connect (a thin integration layer for integrating the Commerce Engine with Sitecore Commerce Connect Core) on the Content Delivery instance, configured for Azure Search.
Sitecore Commerce Engine Connect CD Solr	Contains Commerce Engine Connect (a thin integration layer for integrating the Commerce Engine with Sitecore Commerce Connect Core) on the Content Delivery instance, configured for Solr.
Sitecore Commerce Engine Connect CM Azure	Contains Commerce Engine Connect (a thin integration layer for integrating the Commerce Engine with Sitecore Commerce Connect Core) on the Content Management instance, configured for Azure Search.
Sitecore Commerce Engine Connect CM Solr	Contains Commerce Engine Connect (a thin integration layer for integrating the Commerce Engine with Sitecore Commerce Connect Core) on the Content Management instance, configured for Solr.

Package	Description
Sitecore Commerce Experience Accelerator	Contains Commerce-specific extensions to the Sitecore Experience Accelerator's (SXA) templated UX layouts (for example, UI renderings used to display a catalog in the Storefront).
Sitecore Commerce Experience Accelerator Storefront	Contains the sample and starter storefront as an integrated part of the Sitecore XC solution, and built using Sitecore's new SXA UX layouts.
Sitecore Commerce Experience Accelerator Habitat	Contains the Habitat sample catalog for the Storefront site.
Sitecore Commerce Experience Accelerator Storefront Themes	Contains the themes used for the SXA Storefront site.
Sitecore Commerce ExperienceAnalytics Core	Contains Commerce Experience Analytics installer and core files required to integration with Commerce.
Sitecore Commerce ExperienceProfile Core	Contains Commerce Experience Profile installer and core files required to integrate Experience Profile with Commerce.
Sitecore Commerce Marketing Automation Core	Contains Commerce Marketing Automation installation and core files.
Sitecore Commerce Marketing Automation for Azure	Contains Commerce Marketing automation files required to setup the Marketing Automation Engine service.

2. Preparing your environment

This section outlines the tasks you must complete before installing your Sitecore XC solution on the Azure App Service, and describes the following tasks:

- Set up Azure storage
- Obtain a valid client certificate
- Create a Braintree account

2.1. Set up Azure storage

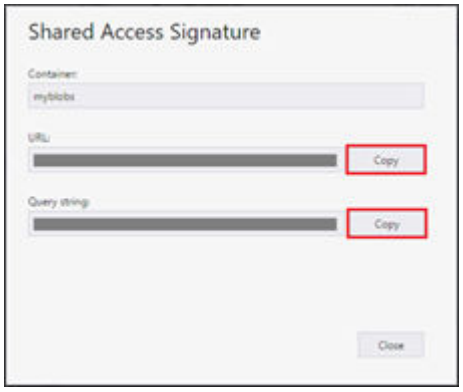
The WDPs containing the application code and resources must be accessible to the Azure Resource Manager over the Internet. You can use a Microsoft Azure® storage account to host the Sitecore WDPs and ARM templates for your deployment.

Follow the instructions on the Microsoft Azure Storage site to create a storage account (with type *Blob storage*) and use the Azure Storage Explorer to create blob containers to store your Sitecore WDPs and ARM templates.

Once you have created your blob containers, create a Shared Access Signature (SAS) token for the storage container. When setting the Start time and Expiry time for the SAS token, make sure that you allow enough time to guarantee access during the scheduled deployment.

After you have created the SAS token, make a note of the following values in the Shared Access Signature dialog:

- **URL**: location to access the WDPs in Azure storage
- **Query string**: contains the SAS token.



NOTE
The URL and SAS token are required for later use in the ARM templates, so for each WDP that you upload, take note of its URL and append the SAS token to it.

2.2. Obtain a valid client certificate

Sitecore XP is designed to be secure by default. This means that the Sitecore roles use the Secure Sockets Layer (SSL) with client certificate validation to communicate with each other.

In addition, the Sitecore Identity Server certificate must contain information to enable a key signing service for its token creation and validation services.

The client certificate validation requires you to set up a client certificate as part of the Sitecore installation process on Azure. You can choose to generate a self-signed certificate to meet this installation requirement or obtain one from a certificate authority.

Before you start your Sitecore XC deployment, you must obtain or generate an authentication certificate and store it in PKCS #12 format (.pfx). For developer environments, you can generate a self-signed certificate using PowerShell. For productions environments, you must obtain a certificate from a certified authority because of potential security concerns.

For more information on client certificate for Sitecore XP deployments, see the topic Client certificate for Sitecore deployments topic on the Sitecore Documentation site.

2.2.1. Generate the certificate

The Sitecore XC Commerce Engine SDK contains a script for generating a self-signed certificate.

Sitecore recommends that you modify the existing certificate generation script and save it with a new name (to preserve a record of the factory default script). After you specify values and paths to reflect your own environment you can run the script to generate the certificate.

To generate the certificate:

1. Go to the Sitecore Experience Commerce Download page and download the Sitecore XC solution package for on-premises (using the **Packages for On Premise** link).
2. Extract the files.
3. Locate the `Sitecore.Commerce.Engine.SDK.x.x.xx` package and extract the contents.
4. Navigate to the `scripts` folder and open the `New-DevelopmentCertificate-Azure.ps1` file in a text editor.
5. Save a copy of the file with a new name (for example, `MyNew-Certificate.ps1`).
6. Specify values for the following parameters, according to your own environment. All other values in the script should remain the same as the specified defaults.
   Descriptions of all parameters in the certificate generation script are available here.

Parameter	Description
`certificateDnsName`	The host name associated with the certificate.
`certificatePassword`	The password for the certificate private key.
`certificateStore`	The store in which to store the new certificate.
`certificateFriendlyName`	A friendly name for the certificate.
`certificateOutputDirectory`	Output directory for the certificate.

7. Add the `-Provider` parameter to the new certificate code block (starting on line 14):

```
$certificate = New-SelfSignedCertificate `
 -Subject $certificateDnsName `
 -DnsName $certificateDnsName `
 -KeyAlgorithm RSA `
 -KeyLength 2048 `
 -NotBefore (Get-Date) `
 -NotAfter (Get-Date).AddYears(1) `
 -CertStoreLocation $certificateStore `
 -FriendlyName $certificateFriendlyName `
 -HashAlgorithm SHA256
 -KeyUsage DigitalSignature, KeyEncipherment, DataEncipherment `
 -Provider "Microsoft Enhanced RSA and AES Cryptographic Provider" `
 -TextExtension @("2.5.29.37={text}1.3.6.1.5.5.7.3.1")
```

8. Save the file and run the script in an elevated PowerShell console.

2.2.2. Convert the PFX file to a Base64 string

You must convert the PFX certificate you generated to a Base64-encoded string so that you can specify the value in the ARM template when you deploy your solution to Azure.

To convert the PFX certificate:

1. In a PowerShell session, run the following command:

```
$fileContentBytes = get-content '<path-to-certificate.pfx>'
-Encoding Byte
[System.Convert]::ToBase64String([System.IO.File]::ReadAllBytes(“path to pfx file”))
```

2. Note the location of the output file. You need to specify the Base64 string in the `azuredeploy.parameters.json` file.

> **NOTE**
> The `authCertificateBlob` parameter in the `azuredeploy.parameters.json` file requires the full Base64 string from step 1, and not the path to the certificate.

2.3. Create a Braintree account

You need a Braintree sandbox account to enable web payment functionality through the Commerce Engine.

To create a Braintree sandbox account:

1. Open a browser and go to https://www.braintreepayments.com/sandbox.
2. Enter the required information in the **Sign up for the sandbox** pane, and click **Try the sandbox**.
3. Follow the instructions to activate and log in to your Braintree sandbox account.

When you receive your Braintree Sandbox account information, note the **Merchant ID**, **Public Key**, and **Private Key** values. You need to specify this information in the ARM template parameters file for your deployment.

3. Deploying from the Azure portal

You can deploy your Sitecore XC solution from the Azure Portal, using the Sitecore Marketplace application. When you deploy your solution from the Azure Portal, you deploy the Sitecore Experience Platform (XP) solution, and include Sitecore XC and the SXA Storefront as modules of that deployment.

To deploy your Sitecore XC solution from the Azure portal:

1. Login to your Microsoft Azure portal.
2. Click on **Create a resource**. In the left pane.
3. In the **Azure Marketplace** window, search for Sitecore Experience Cloud.
4. On the Sitecore Experience Cloud tab, click **Create**.
5. Select your Azure subscription.
6. Under Resource Group, select the **Create new** option and specify a name for the new resource group. Note that currently, all characters must be lower case.
7. Under Sitecore Settings, click on **Configure required settings** to open the settings panel.
8. On the **Environment** tab, select values for the following parameters from each pre-populated list:
    - **Version** (select the current version of Sitecore XC)
    - **Topology** (select “Sitecore Experience Cloud XP”)
    - **Size** (select “Small”, “Medium”, or “Large”)
    - **Additional Modules** (select “Sitecore Experience Commerce 9.1", and optionally, the corresponding Sitecore Experience Accelerator Storefront)

    > **NOTE**
    > When you select the Sitecore Experience Accelerator Storefront module, the Sitecore Experience Commerce 9.1 and Sitecore Experience Accelerator 1.8 modules are automatically selected as well.

9. Click **Next**.
10. On the **Region** tab, select a region for the deployment location and the Application Insights region from the drop-down lists, and click **Next**.
11. On the **Credentials** tab, specify values for the following:
    - Sitecore Administrator password
    - SQL Server login and password
    - License XML file (upload to the portal)
    - Authentication certificate (either auto-generate a self-signed certificate or use an existing certificate from a trusted authority), with name and password
    - Braintree account information (optional)

- Environment name
- Shop name, currency and language

> **NOTE**
> If the value for the shop currency is not one of the default currencies supported by the Commerce Engine (USD, CAD, or EURO), you must add the currency to the **Default** currency set on the Commerce Control Panel after you deploy your solution to the Azure App service.

12. Click **Next**.
13. On the **Summary** tab, review the information and click **Ok**.
14. Review the Legal terms and click **Create** to start the deployment.

You can monitor your deployment by going to your **Resource groups** page and checking the status of your new resource group.

Deployment takes approximately 2 hours. When deployment is done, you must complete the tasks described in the Post-installation steps topic.

4. Deploying using the Sitecore Azure Toolkit

You can deploy your Sitecore XC solution using the Sitecore Azure Toolkit.

When you deploy your solution using the Sitecore Azure Toolkit, you upload the solution Web Deploy Packages (WDPs) into Azure storage, customize the Azure Resource Manager (ARM) template, and deploy the solution to Azure from a PowerShell session.

This chapter contains the following sections:

4.1. Install the Sitecore Azure Toolkit

The Sitecore Azure Toolkit contains the tools and resources necessary to prepare and deploy Sitecore solutions to the Microsoft Azure App Service®.

Download and install the Sitecore Azure Toolkit, as described in the Getting started with Sitecore Azure Toolkit topic.

4.2. Obtain the Web Deploy Packages

Sitecore creates WDPs from standard Sitecore deployment packages during the packaging process that have been modified to run on Azure during the packaging process. Use these WDPs for a standard original deployment of Sitecore.

You can download pre-built WDPs from the Sitecore Download page. The WDP(s) come in a single .zip file and are grouped per version and per topology. After you download the WDPs, you must extract them from the .zip file before the Sitecore Azure Toolkit can use them.

The following table lists the required WDPs for your Sitecore XC 9.1 solution to Azure. All WDP files have an extension of `.scwdp.zip`.

> **NOTE**
> For Sitecore XC 9.1, only the Sitecore XP Scaled topology is supported.

Product	Web Deploy Packages
Sitecore XP 9.1.1 Available from the Sitecore Experience Platform Download page.	Sitecore 9.1.1 rev.xxxxx (Cloud)_cd Sitecore 9.1.1 rev.xxxxx (Cloud)_cm Sitecore 9.1.1 rev.xxxxx (Cloud)_dds Sitecore 9.1.1 rev.xxxxx (Cloud)_prc Sitecore 9.1.1 rev.xxxxx (Cloud)_rep Sitecore 9.1.1 rev.xxxxx (Cloud)_xp1collection Sitecore 9.1.1 rev.xxxxx (Cloud)_xp1collectionsearch Sitecore 9.1.1 rev.xxxxx (Cloud)_xp1collectionsearchSolr Sitecore 9.1.1 rev.xxxxx (Cloud)_xp1cortexprocessing Sitecore 9.1.1 rev.xxxxx (Cloud)_xp1cortexreporting Sitecore 9.1.1 rev.xxxxx (Cloud)_xp1marketingautomation Sitecore 9.1.1 rev.xxxxx (Cloud)_xp1marketingautomationreporting Sitecore 9.1.1 rev.xxxxx (Cloud)_xp1referencedata Sitecore.Patch.EXM (Cloud)_CM.zip

Product	Web Deploy Packages
Sitecore XC 9.1 Available from the Sitecore Experience Commerce Download page.	Adventure Works Images Sitecore Commerce Connect Core xx.x.xx Sitecore Commerce Connect Schema Definitions for IndexWorker xx.x.xx Sitecore Commerce Connect Definitions for xConnect xx.x.xx Sitecore Commerce Experience Accelerator x.x.xxx Sitecore Commerce Experience Accelerator Habitat Catalog x.x.xxx Sitecore Commerce Experience Accelerator Storefront x.x.xxx Sitecore Commerce Experience Accelerator Storefront Themes x.x.xxx Sitecore Commerce ExperienceAnalytics Core xx.x.xx Sitecore Commerce ExperienceProfile Core xx.x.xx Sitecore Commerce Marketing Automation Core xx.x.xx Sitecore Commerce Marketing Automation for Azure xx.x.xx Sitecore.BizFx.x.x.x Sitecore.Commerce.Engine.Azure.x.x.xx Sitecore.Commerce.Engine.Connect.CD.Azure.x.x.xx Sitecore.Commerce.Engine.Connect.CD.Solr.x.x.xx Sitecore.Commerce.Engine.Connect.CM.Azure.x.x.xx Sitecore.Commerce.Engine.Connect.CM.Solr.x.x.xx Sitecore.Commerce.Engine.Solr.x.x.xx Sitecore.Commerce.Habitat.Images-x.x.x Sitecore.Identity.Config.Commerce.x.x.x
SXA 1.8.1 Available from the Sitecore Experience Accelerator Download page.	Sitecore Experience Accelerator WDP for 9.1 Sitecore Experience Accelerator CD WDP for 9.1 Sitecore PowerShell Extensions WDP 5.0 for Sitecore 9
Sitecore Azure Toolkit Bootloader Available from the *Addons* folder in the Sitecore Azure Toolkit zip file.	Sitecore.Cloud.Integration.Bootload.wdp.zip

4.2.1. Download the WDPs

For each set of WDPs you need, perform the following steps:

1. Go to the appropriate Sitecore Download page (as listed in the table above).
2. Scroll to the **Download options for Azure AppService** section and click on the appropriate link for your Sitecore product.
3. When you have downloaded the zip file, unzip the file to unpack the individual WDP files.

4.2.2. Upload the WDPs to Azure storage

Your Sitecore WDPs must be accessible over the Internet to the Azure Resource Manager. These instructions assume that you are using Azure Storage to host the WDPs.

When you upload your WDPs to an Azure storage container, you create a Shared Access Signature (SAS) token for the storage container. The SAS allows temporary access to the WDPs during the deployment process.

For each set of downloaded WDPs, perform the following steps:

1. Open **Azure Storage Explorer** and connect to a Microsoft Azure storage account.
2. Upload the WDPs for your configuration to the container that you created earlier.

> **NOTE**
> Make sure that you take note of the URL for each WDP that you upload, and append the SAS token that you generated earlier to it. You must specify this URL (with SAS token) in the ARM templates.

4.3. Configure the ARM template

ARM templates are distributed on Github and provide a description of Sitecore environments hosted on Microsoft Azure App Service. The templates are compatible with the WDPs available on dev.sitecore.net or those produced by the Sitecore Azure Toolkit.

The structure and organization of the templates are nested. The main template (`azuredeploy.json`) references a parameters file (`azuredeploy.parameters.json`), an infrastructure template and an application template. The deployment responsibilities are then split respectively between templates to setup resources and templates to install the Sitecore application.

4.3.1. Obtain the ARM template and parameters file

When you deploy to Azure, you must specify an ARM template, which contains the instructions for the Azure deployment, and include environment-specific parameter values in an Azure environment file.

For this scenario, you use the Sitecore XP 9.1 ARM template to deploy all required software and modules to the Azure App Service.

To obtain the ARM template and parameters file:

1. Go to the Sitecore GitHub repository and navigate to the `Sitecore 9.1.1\XP` folder.
2. In the `Sitecore 9.1.1\XP` folder, click on the `azuredeploy.json` file to open it.
   The file opens in the browser window.
3. In the `azuredeploy.json` file, click on **Raw** in the document header row.
   The file opens in a new browser window.
4. Copy the URL directly from the address bar and make a note of the information for use in the Azure environment file.
5. Return to the `Sitecore 9.1.1\XP` folder and download the `azuredeploy.parameters.json` file.

> **NOTE**
> You only need to download the Sitecore 9.1.1 XP version of the `azuredeploy.parameters.json` file. This environment configuration file specifies parameters for all of the components of your Azure deployment (including Sitecore XC and SXA Storefront).

4.3.2. Customize the Azure environment template

You specify values for your deployment environment in the Sitecore XP `azuredeploy.parameters.json` file. The ARM template uses the parameters file to obtain environment-specific values during deployment.

By adding modules to the `azuredeploy.parameters.json` file, you can instruct the ARM template to deploy Sitecore XC and the SXA Storefront as part of the Sitecore XP deployment to Azure.

To customize the Azure environment template:

1. Open the `azuredeploy.parameters.json` file and specify values for the parameters listed below. These parameters are not Sitecore XC-specific, but are required for the Sitecore XP deployment to Azure.

Parameter	Description
`deploymentID`	The unique identifier for the deployment. This string must be all lower-case characters, contain no spaces, and not be longer than 64 characters.
`location`	The geographical region of the current deployment.
`sitecoreAdminPassword`	Password for the *sitecore\admin* user when Sitecore is deployed.
`licenseXml`	Path to the Sitecore license.xml file.
`repAuthenticationApiKey`	A unique value, for example, a GUID, which is used to authenticate when communicating from Content Management role to the Reporting Web App. The minimum length required is 32 characters.
`sqlServerLogin`	Name of the administrator account for the Azure SQL Server (for core, master, web, and reporting SQL Azure databases).
`sqlServerPassword`	Password for the administrator account for the Azure SQL server.
`siMsDeployPackageUrl`	The HTTP(s) UrL to a Sitecore Identity Server WDP.
`cmMsDeployPackageUrl`	The URL to a Sitecore XM Content Management WDP.
`cdMsDeployPackageUrl`	The URL for a Sitecore XM Content Delivery WDP.
`prcMsDeployPackageUrl`	The URL for a Sitecore XP Processing WDP.
`repMsDeployPackageUrl`	The URL for a Sitecore XP Reporting WDP.
`xcRefDataMsDeployPackageUrl`	The URL for an xConnect Reference Data service WDP.
`xcCollectMsDeployPackageUrl`	The HTTP(s) URL for an xConnect Collection service WDP.
`xcSearchMsDeployPackageUrl`	The HTTP(s) URL for an xConnect Collection Search service WDP.
`cortexProcessingMsDeployPackageUrl`	The HTTP(s) URL for a Cortex Processing service WDP.
`cortexReportingMsDeployPackageUrl`	The HTTP(s) URL for a Cortex Reporting service WDP
`maOpsMsDeployPackageUrl`	The HTTP(s) URL for a Marketing Automation service WDP.
`maRepMsDeployPackageUrl`	The HTTP(s) URL for a Marketing Automation Reporting service WDP.
`authCertificateBlob`	The Base64-encoded blob of the authentication certificate for the Sitecore XC solution.
`authCertificatePassword`	The password for the authentication certificate.

2. If you are using SolrCloud as a search provider, add the following parameter to specify the URL for the SolrCloud instance:

```
"solrConnectionString" : <URL for SolrCloud instance>
```

3. If you are using self-signed certificates, you must add the following parameter to force xConnect to accept self-signed certificates:

```
"allowInvalidClientCertificates": {
    "value": true
```

4. Add a `modules` block after the last parameter in the file using the following code snippet:

```
"modules": {
  "value": {
    "items": [
       /* add module entries here */
```

```
        ]
      }
    }
```

5. Add the Commerce module to the modules block to specify the location of the Sitecore XC ARM template, the location of the Sitecore XC WDPs, and Braintree account information (optional).

```
"modules": {
  "value": {
    "items": [
      {
       "name": "Commerce",
       "templateLink": "<URL for the Sitecore XC ARM template>",
       "parameters": {
       "templateLinkAccessToken": "<SAS-token>",
       "commerceConnectMsDeployPackageUrl": "<URL for WDP>",
       "commerceEngineAzureMsDeployPackageUrl": "<URL for WDP>",
       "commerceEngineSolrCloudMsDeployPackageUrl": "<URL for WDP>",
       "commerceEngineConnectCdAzureMsDeployPackageUrl": "<URL for WDP>",
       "commerceEngineConnectCmAzureMsDeployPackageUrl": "<URL for WDP>",
       "commerceEngineConnectCdSolrMsDeployPackageUrl": "<URL for WDP>",
       "commerceEngineConnectCmSolrMsDeployPackageUrl": "<URL for WDP>",
       "identityConfigCommerceMsDeployPackageUrl": "<URL for WDP>",
       "bizfxCloudMsDeployPackageUrl": "<URL for WDP>",
       "adventureWorksImagesDeployPackageUrl": "<URL for WDP>",
       "habitatImagesDeployPackageUrl": "<URL for WDP>",
       "indexWorkerDefinitionsDeployPackageUrl": "<URL for WDP>",
       "xConnectDefinitionsDeployPackageUrl": "<URL for WDP>",
       "xAnalyticsCoreDeployPackageUrl": "<URL for WDP>",
       "xProfileCoreDeployPackageUrl": "<URL for WDP>",
       "maCoreDeployPackageUrl": "<URL for WDP>",
       "maAzureDeployPackageUrl": "<URL for WDP>",
       "braintreeMerchantId": "<merchant ID for Braintree account>",
       "braintreePublicKey": "<public key for Braintree account>",
       "braintreePrivateKey": "<private key for Braintree account>",
       "braintreeEnvironment": "<Braintree environment>",
       "environmentName=": "<environment name>"
       "defaultShopName": "<default shop name>",
       "defaultShopCurrency": "<default shop currency>",
       "defaultShopLanguage": "<default shop language>",
      }
    },
```

> **NOTE**
> If the value for `defaultShopCurrency` is not one of the default currencies supported by the Commerce Engine (USD, CAD, or EURO), you must add the currency to the **Default** currency set on the Commerce Control Panel after you deploy your solution to the Azure App service.

6. Add the SXA and SXA Storefront modules to the modules block, to specify the location of the SXA ARM template, and the location of the SXA and Storefront WDPs:

```
        {
         "name": "sxa",
         "templateLink": "<URL for SXA ARM template>",
         "parameters": {
         "cdSxaMsDeployPackageUrl": "<URL for SXA CD WDP>",
         "cmSxaMsDeployPackageUrl": "<URL for SXA CM WDP>",
         "speMsDeployPackageUrl": "<URL for Powershell Extensions WDP>"
         "solrSupportSxaMsDeployPackageUrl: "<URL for SXA CM WDP>",
         "templateLinkAccessToken": "<Token>"
        }
    },
        {
         "name": "sxa-sf",
         "templateLink": "<URL for SXA Storefront ARM template>",
         "parameters": {
         "sxaMsDeployPackageUrl": "<URL for WDP>",
         "sxaStorefrontMsDeployPackageUrl": "<URL for WDP>",
         "sxaHabitatCatalogMsDeployPackageUrl": "<URL for WDP>",
         "sxaStorefrontThemesMsDeployPackageUrl": "<URL for WDP>",
         "environmentName"": "<environment name>",
         "siteName": "<site name>"
        }
     },
```

NOTE
If you are not using Solr as the Search provider, the value of the `solrSupportSxaMsDeployPackageUrl` parameter can be any string longer than 8 characters.

7. Add the Bootloader module to the modules block, to specify the location of the Bootloader ARM template and the location of the Bootloader WDP:

```
        {
         "name": "bootloader",
         "templateLink": "",
         "parameters": {
         "msDeployPackageUrl": ""
        }
    },
```

8. Save your changes to the file.

4.4. Deploy the solution to Azure

After you have successfully uploaded the Sitecore WDPs and configured the ARM template with the values necessary for your environment, you can deploy your solution to Azure.

When you are ready to deploy your Sitecore solution to Azure, do the following:

1. Open an elevated PowerShell window and navigate to the `Sitecore Azure Toolkit` folder.
2. Add the following Azure account to your PowerShell session:

   ```
   Add-AzureRMAccount
   ```

   The system prompts you to login to your Azure Subscription account.
3. Load the Sitecore Azure Toolkit module:

   ```
   Import-Module .\tools\Sitecore.Cloud.Cmdlets.psm1
   ```

4. Select the subscription that you want to deploy to:

   ```
   Set-AzureRMContext -SubscriptionName "<name of the subscription>"
   ```

5. Enter the following command to initialize deployment:

   ```
   Start-SitecoreAzureDeployment -location <String> -Name <String> -
   ArmTemplateUrl <String> -ArmParametersPath <String> -LicenseXmlPath
   <String>
   ```

   The following table describes the parameters accepted by the `Start-SitecoreAzureDeployment` cmdlet:

Parameter	Description
`Location`	The geographical region of the current deployment.
`Name`	The name of the resource group for the new environment. It can refer to a new or an existing resource group, and is usually the same as the deployment ID.
`ArmTemplateURL`	The URL of the Sitecore XP ARM template file. It points to the HTTP(S) location that is hosting the templates.
`ArmParametersPath`	The path to the customized `azuredeploy.parameters.json` file.
`LicenseXmlPath`	The path to the Sitecore license file that you want to deploy to the environment.

5. Post-installation steps

Once you have successfully installed the Sitecore XC software, you must complete the following tasks to complete your deployment:

- Bootstrap and initialize the Commerce Engine
- Generate catalog templates
- Configure user accounts
- Enable Marketing Automation
- Create an SXA Storefront tenant and site
- Republish the site and rebuild the search indexes
- Enable antiforgery

5.1. Bootstrap and initialize the Commerce Engine

After you have deployed the Sitecore XC solution to Azure, you must log into the Sitecore Content Management (CM) instance to initialize Sitecore services, then run the Sitecore XC bootstrap operation to ensure that all settings are written to the global database.

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.

The following instructions assume that you have installed the Postman application, imported the sample collections from the Sitecore Commerce Engine SDK, configured your environment, and retrieved a bearer token to access the Commerce Engine API, as described here.

> **NOTE**
> When you place a call to the Commerce Engine API from outside the Commerce Business Tools (for example, using Postman), you must disable the anti-forgery protection setting in the `wwwroot\config.json` file.

To run the bootstrap and initialize operations:

1. In the Postman **Collections** pane, navigate to the *SitecoreCommerce_DevOps* folder.
2. Open the *1 Environment Bootstrap* folder, and execute the **Bootstrap Sitecore Commerce** call.
3. In the top right corner of Postman, click the environment selector, and select **Habitat Environment**.
4. Open the *3 Environment Initialize* folder, and execute the **Ensure\Sync default content paths** call.
5. In the *3 Environment Initialize* folder, execute the **Initialize Environment** call.
6. In the top right corner of the Postman application, click the environment selector, and click **AdventureWorks Environment**.
7. Repeat step 4 and step 5.

After you have run the bootstrap and initialize operations, go to the Azure Portal and restart the Minions service.

5.2. 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 open the **Sitecore Launchpad** (e.g., https://<server>/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)**.**

5.3. 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 the pre-defined roles and associated permissions for the Sitecore XC Business Tools.

5.4. Enable Marketing Automation

Sitecore Marketing Automation (MA) campaign functionality is disabled by default in the Sitecore XC deployment.

> **NOTE**
> You must perform these steps to ensure that the Storefront site functions properly.

To enable Marketing Automation and Live Events:

1. Enable the xConnect collection role on the Marketing Automation Operations server, as described in this Knowledge Base article.
2. Create missing stored procedures, as described in this Knowledge Base article.
3. Go to the `<ma_ops>/wwwroot/app_config` folder and add the `xdb.processing.pools` to the `ConnectionStrings.config` file for the Marketing Automation Operations role:

   ```
   <add name="xdb.processing.pools" connectionString="user
   id=poolsuser;password=ODPYFLX7R2D3Qkzpbw2jdctry6KZPBW2JDCTRY64@;datasourc
   e=xc902rc4a180705-sql.database.windows.net;Initial
   Catalog=xc902rc4a180705-poolsdb"/>
   ```

   You can copy the entry from the `<xc_collect>/wwwroot/app_config/ConnectionStrings.config` file to make sure you are using the proper database user id.

5.5. Create an SXA Storefront tenant and site

If you want to use the SXA Storefront site as a starting point to create your own site, you must create a new tenant and site using the Commerce Storefront Template.

1. Open a browser, and open the **Sitecore Launchpad** (e.g., https://<server>/sitecore).
2. Navigate to the **Content Editor** and create a new tenant and site with commerce features as described here.

> **NOTE**
> To install the sample site, you must select the *Commerce Storefront Template* feature when you run the **Create a new Experience Accelerator site** wizard.

5.5.1. Add the Storefront domain to the domains.config file

When you create a new storefront using the **Create a new Experience Accelerator site** wizard, Sitecore creates a corresponding new security domain in the `/App_Config/security/domains.config` file on a Content Management (CM) instance.

In distributed environments like Azure, you must manually update the `domains.config` file, which contains the definition of security domains, on all other instances.

Copy the file from the CM instance where you created the Storefront(s) to any other CM instance(s) and the CD instance(s). On Azure, you can copy files from one instance to another using the App Service Editor.

5.6. Republish the site and rebuild the search indexes

You must re-index the master and web indexes for the Sitecore XC site. You can perform both of these operations from the **Control panel** on the **Sitecore Launchpad**.

> **NOTE**
> It is best practice to rebuild each index separately. It does not matter which one you choose to build first.

To re-publish the site:

1. Log in to Sitecore Launchpad and click on **Content Editor**.
2. Click on the **Publish** tab.
3. Click **Publish** and select **Publish site** from the drop-down list.
4. In the **Publish Site** window, select **Republish – publish everything** and click **Publish**.

To re-index the Search indexes:

1. In the Control Panel, click on **Indexing Manager**.
2. In the **Indexing Manager** dialog box, select **sitecore_master_index** and click **Rebuild**.
3. When the Sitecore master index is rebuilt, repeat the steps above to rebuild the **sitecore_web_index**.

5.7. Enable antiforgery

Enable antiforgery to create antiforgery tokens that protect Sitecore Experience Commerce (XC) applications against cross-site request forgery (CSRF).

> **NOTE**
> This procedure assumes you have already setup your own custom domain for Azure App Service. Due to browser restrictions, you cannot use the *azurewebsites.net* domain to enable antiforgery protection.

When you enable antiforgery for a specific domain, Sitecore XC applies the protection to its sub-domains.

To enable antiforgery on a domain and its sub-domains:

1. Open the Commerce Engine `config.json` (for example, `c:\inetpub\wwwroot\<CommerceAuthoring_SC9>\wwwroot\config.json`).
2. Set `"AntiForegeryEnabled"` to `true`.
3. Set `"CommerceServicesHostPostfix"` to specify `<your domain>`.

The following example shows a configuration that creates antiforgery tokens for the domain **.yoursite.com* (and its sub-domains). This approach allows each website, for example, each Commerce Engine BizFx website, to have its own distinct sub-domain with antiforgery protection enabled.

```
"AntiForegeryEnabled": true
"CommerceServicesHostPostfix": "yoursite.com"
```