# Sitecore Azure Toolkit 1.0

*All the official Sitecore documentation.*

sitecore®
Own the experience™

# Sitecore Azure App Service overview

From Sitecore version 8.2 Update-1 onwards, the Sitecore Experience Platform supports the Microsoft Azure App Service®. This means that you are now able to deploy scalable Sitecore solutions on the modern Microsoft Azure® PaaS infrastructure.

This topic describes:


- New features and integrations
- Comparison to the standard Sitecore Azure module
- Default environments
- The Sitecore Azure Toolkit


## New features and integrations

When running on the Azure App Service, the Sitecore Experience Platform integrates tightly with Azure services, for example:

- Content databases and the xDB Reporting database use the Microsoft Azure SQL® service.
- *Content Search* and the *Contact Segmentation* index use the Sitecore Azure Search provider to use the Microsoft Azure Search® service.
- Sitecore integrates with Microsoft Application Insights® to collect logs and information about performance counters. Logs are also written to system trace. This means logs can be collected by the Microsoft App Service infrastructure or monitored with either the Microsoft App Service Editor® or with Microsoft Visual Studio® using the Azure SDK.
- The Session state is managed using the Microsoft Azure Redis Cache service and the Sitecore session state provider for Redis.
- Sitecore instances are security-hardened by default, this means:
  - The Sitecore password is set during provisioning.
  - The *Content Management* (CM), *Processing*, and *Reporting* roles can only be accessed by HTTPS.
  - Access to the Sitecore shell is locked down on *Content Delivery* (CD) instances.
  - Unauthorized access to Sitecore shell resources is locked down on all instances.
  - It is possible to configure IP filtering when provisioning the environment.

## Comparison to the Sitecore Azure Module

Using the Sitecore Azure module for Sitecore versions 8.0, and 8.1 means *CM* and *CD* instances can run using Microsoft Azure Cloud Services. The Sitecore Azure App Service provides the following benefits compared to the standard Sitecore Azure module:

- Support for hosting full Experience Management (XM) or Sitecore Experience Platform (XP) environments in Microsoft Azure, without having to run an on-premise management instance.
- Simpler deployment and configuration management with more supported tools, such as: Microsoft Azure Resource Manager (ARM), Microsoft Web Deploy, FTP, GIT, and the Sitecore Package Installation Wizard.
- More insight into Sitecore diagnostics and performance with Microsoft Application Insights®.
- Reduced cost and better scalability of Microsoft Azure Websites, (compared to Cloud Services).

## Default environments

The Microsoft Azure SDK provides ARM templates for the following environments by default:

- XP0 is a single-instance Sitecore XP installation for development and testing purposes. This configuration is not security-hardened and Sitecore recommends using XM1 or XP1 environments for production deployments.
- XM1 is a scaled XM environment with *CM* and *CD* roles.
- XP1 is a scaled XP environment with *CM*, *CD*, *Processing*, and *Reporting* roles.

## The Sitecore Azure Toolkit

Sitecore provides a number of tools and resources that simplify deployment of Sitecore solutions to Azure. The toolkit includes:

- Prebuilt Web Deploy packages for deployment to Azure. The packages include the necessary role configuration and integration with Azure services.
- Microsoft ARM templates that describe Sitecore environments used with the Azure Portal, Microsoft Azure Powershell®, or the Microsoft Azure RM API.
- The Microsoft Azure Powershell module that contains commandlets for deploying and packaging Sitecore solutions and modules.
- Version-specific resources that are used while packaging Sitecore solutions.

Send feedback about the documentation to docsite@sitecore.net.

# Configure the Bootloader module for a Sitecore deployment

The Bootloader module is a tiny module that facilitates the installation of supported Sitecore modules. You must always add the Bootloader module to the `modules` parameter of your Sitecore App Service deployment when installing other modules.

To configure the Bootloader module:

1. Locate the WebDeploy package (WDP), which is the Sitecore.Cloud.Integration.Bootload.wdp.zip) of the Bootloader module in `resources\<version to be deployed>\addons` folder of your Sitecore Azure Toolkit installation.
2. Upload the WDP to the storage account and note the URL of the package because you will need to add it to the snippet in step 4. If you are running Sitecore 8.2.7, then in step four change the parameter value to: `bootloaderMsDeployPackageUrl`

Note

Sitecore version 8.2.7 uses an ARM template that has already configured the bootloader by default, so if you are running Sitecore 8.2.7 ignore the following steps, (3, 4, and 5),

1. On Github, in the addons folder of your Sitecore version and environment configuration, locate the bootloader.json template.
2. Add the following snippet to the `modules` parameter of your `azuredeploy.parameters.json` file:

```
{
```

```
"modules": {
"value": {
"items": [
{
"name": "bootloader",
"templateLink" : "<link to the Bootloader template>",
"parameters": {
"msDeployPackageUrl" : "<link to the Bootloader WDP>"
}
}
]
}
}
}
```

3. Populate the parameters for the Bootloader module:
   - For `templateLink`, go to Github and use the URL of the bootloader.json template for your particular topology. You can also upload the template to your storage account.
   - For `msDeployPackageUrl`, use the URL of the WDP package for the Bootloader module.

Send feedback about the documentation to docsite@sitecore.net.

# Deploy a new Sitecore environment to Azure App Service

You use the Sitecore Azure Provisioning Toolkit to deploy a new Sitecore environment to Microsoft Azure App Service®. This topic describes how to:

- Plan your environment
- Prepare WebDeploy packages
- Obtain MongoDB connection strings
- Download and configure an environment template
- Invoke the deployment command

## Plan your environment

Sitecore Azure Toolkit by default supports three Sitecore configurations to suit different needs:

| Configuration | Description |
|---|---|
| XP0 | This is Sitecore Experience Platform running as a single WebApp instance. Use this configuration for development and testing purposes. It is best practice to use the XM1 or XP1 configuration in production environments for security and scalability reasons. |
| XM1 | This is Sitecore Experience Management configuration running two roles: Content Delivery and Content Management. Use this environment when you are not planning to use Analytics and Marketing features of Sitecore Experience Platform (that is, in CMS-only mode). |
| XP1 | This is Sitecore Experience Platform configuration running four roles: *Content Delivery*, *Content Management*, *Processing*, and *Reporting*. Use this environment when you are planning a fully-featured Sitecore Experience Platform installation. |

## Prepare WebDeploy packages

Azure Resource Manager (ARM) requires WebDeploy packages containing the application code and resources to be available for download over the Internet. One option to host WebDeploy packages is to create a Microsoft Azure® storage account.

To prepare WebDeploy packages:

1. Obtain a Microsoft Azure subscription to deploy a Sitecore environment.
2. Create a Microsoft Azure storage account using this guide.
3. Use Azure Storage Explorer to connect to a Microsoft Azure storage account and upload Sitecore WebDeploy packages for the chosen configuration. Ensure you use a *blob* type storage container.
   - You can download prebuilt WebDeploy packages from the Sitecore Experience Platform Download page.

   Note

   Webdeploy packages have the extension .scwdp.zip and contain the role name as part of the file name, for example, Sitecore 8.2 rev. 161115_cm.scwdp.zip.
4. To obtain URLs for your WebDeploy packages, in Azure Storage Explorer, click Copy URL.
5. Use Azure Storage Explorer to create a Shared Access Signature (SAS) token for the storage container and append the token to the WebDeploy package URLs.
6. Make a note of the package URLs for later use in ARM templates.

## Obtain MongoDB connection strings

When deploying XP configurations, you must provide MongoDB connection strings for the analytics database and for three tracking databases. You can choose to host MongoDB with a PaaS service such as MongoDB Atlas or mLab, or you can self-host MongoDB using Azure Virtual Machines. When the MongoDB cluster is ready, note the connection strings for the databases for later use in ARM templates.

Important

If the MongoDB driver attempts to reuse connection strings that Azure has already closed, this may cause connection issues. To avoid this, set the `maxIdleTimeMS` parameter to `60000` for all MongoDB connection strings, for example:

```
<add name="analytics" connectionString="mongodb://user:password@host/analytics?replicaSet=rs1&ssl=true&maxIdleTimeMS=60000" />
```

## Download and configure an environment template

To download and configure the environment template for your selected Sitecore configuration:

1. In the Github repository, for the selected Sitecore configuration, download the environment template (*azuredeploy.json*) and the corresponding parameters file (`azuredeploy.parameters.json`).
2. Open the `azuredeploy.parameters.json` file in the editor and fill in the following parameters:

| Parameter name | Configuration | Description | Value restrictions |
|---|---|---|---|
| `deploymentId` | All | Unique ID of the deployment. This value is prepended to the names of all resources | Must contain only alphanumeric characters and hyphens. Maximum length is 60 characters. |
| `cm.msdeploy.packageurl`<br>`cd.msdeploy.packageurl` | XM1 and XP1 | Links to the WebDeploy packages for the roles. | The URLs must be accessible from Microsoft Azure data centers. Use SAS tokens or an equivalent to provide secure access to the resources. |
| `prc.msdeploy.packageurl`<br>`rep.msdeploy.packageurl` | XP1 only | | |
| `single.msdeploy.packageurl` | XP0 only | | |
| `sitecore.admin.password` | All | Password for 'sitecore\admin' user when Sitecore is deployed. | Minimum length is 8 characters. |
| `sqlserver.login` | All | Name of the administrator user for the virtual SQL Server for core, master, web, and reporting SQL Azure databases. | |
| `sqlserver.password` | All | Password for the administrator user for the virtual SQL Server for core, master, web, and reporting SQL Azure databases. | Minimum length is 8 characters. |
| `rep.authentication.apikey` | XP1 only | Shared authentication key for XDB Reporting service. | Minimum length is 32 characters, for example a GUID. |
| `analytics.mongodb.connectionstring`<br>`tracking.live.mongodb.connectionstring`<br>`tracking.history.mongodb.connectionstring`<br>`tracking.contact.mongodb.connectionstring` | XP0 and XP1 | MongoDB connection strings for analytics and tracking databases. | |
| `applicationinsights.location` | All | Optional location for Application Insights telemetry data storage. East US by default. | The Microsoft Azure Service Availability by Region lists the locations supporting Application Insights. |

## Invoke the deployment command

To invoke the PowerShell command and initiate provisioning:

1. In PowerShell, navigate to the *Azure Toolkit* folder and load the Azure Toolkit module: `Import-Module .\tools\Sitecore.Cloud.Cmdlets.psm1 -Verbose`.
2. Add an Azure account to your PowerShell session: `Add-AzureRMAccount`.
3. If there is access to multiple subscriptions, select the subscription to deploy into: `Set-AzureRMContext -SubscriptionName "<name of the subscription>"`.
4. Start provisioning using the `Start-SitecoreAzureDeployment` commandlet:

```
Start-SitecoreAzureDeployment [-location] <String> [-Name] <String> [-ArmTemplatePath] <String> [-ArmParametersPath] <String> [-L
```

The `Sitecore-SitecoreAzureDeployment` commandlet accepts the following parameters:

| Parameter | Description |
|---|---|
| Location | Name of the Azure datacenter where you want the resources deployed.<br><br>Refer to the Azure data center compatibility table for the list of Microsoft Azure data centers that Sitecore Experience Platform supports deployment to. |
| Name | Name of the resource group for the new environment. It can refer to a new or an existing resource group. This should usually be the same as the deployment ID. |
| ArmTemplatePath | Path to the ARM template file for the environment configuration to deploy. |
| ArmParametersPath | Path to the populated `parameters.json` file for the selected template. |
| LicenseXmlPath | Path to the Sitecore license file to deploy to the environment. |
| LicenseXml | Content of the Sitecore license file (XML). Set this parameter by passing the path of the license file to the `LicenseXmlPath` parameter of the `Start-SitecoreAzureProvisioning` commandlet. |
| SetKeyValue | Use this parameter in a script to deploy several environments by using a common set of default parameter values in `azuredeploy.parameters.json` and supplying environment-specific parameter values in the command line.<br><br>The value of this parameter is a hashtable that contains a subset of parameters declared in `azuredeploy.parameters.json`.<br><br>Any values that you specify in the command line in this parameter will override the values specified in `azuredeploy.parameters.json`.<br><br>By default, the value of this parameter should be an empty hashtable: `-SetKeyValue @{}`.<br><br>For example, to set or override the value in the `deploymentId` parameter, enter the following: `-SetKeyValue @{ "deploymentId" = "<new value>" }`. |

Send feedback about the documentation to docsite@sitecore.net.

# Getting started with the Sitecore Azure Toolkit

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

The Sitecore Azure Toolkit simplifies the task of preparing and deploying a Sitecore solution with:

- PowerShell commandlets to package a Sitecore instance into role-specific packages
- Out-of-the-box integration with Microsoft Azure services: Microsoft Azure SQL®, Microsoft Azure Redis Cache, Microsoft Application Insights®, Microsoft Azure Search®
- Prebuilt ARM templates for frequently used topologies: XM, XP, XPSingle
- Configuration tweaks to run Sitecore on the Azure App Service
- Security features: HTTPS, low-privileged SQL access, secure Sitecore password

This topic describes:


- Deploying Sitecore onto Azure with the Sitecore Azure Toolkit
- Prerequisites
- Downloading the Sitecore Azure Toolkit
- Installing the Sitecore Azure Toolkit
- Azure Resource Manager templates
- Web Deployment Packages
- Using the Sitecore Azure Toolkit


## Deploying Sitecore onto Azure with the Sitecore Azure Toolkit

The following is a high-level overview of the process that takes place when you use the Sitecore Azure Toolkit (SAT) to deploy Sitecore to Azure:

- First, you can download or create Web Deploy packages (WDPs) with the Sitecore Azure Toolkit. WDPs contain the application code and resources required to deploy Sitecore as well as any associated modules (if required). During the packaging process, the content of the WDPs are transformed to ensure correct operation when they are deployed to Azure. These transformations are defined via Sitecore Cargo Payload (SCCPL) packages which SAT references during the packaging process.
- After obtaining the packages, you prepare the Azure Resource Manager templates (ARM templates). These templates instruct the Azure Resource Manager to deploy resources to Azure, this includes both infrastructure (such as Web Apps) and application (WDPs). Sitecore provides default templates, to prepare them means choosing the right templates and providing the right parameters to those templates to configure the required deployment correctly.
- When you have prepared the WDPs and templates, you can deploy all of the required infrastructure and Sitecore applications onto Azure with a single command from SAT.

## Prerequisites

To use the Sitecore Azure Toolkit, ensure you have the following:

- (If you are deploying a version of Sitecore that is earlier than 9.0) A Cloud-hosted MongoDB cluster for the *xDB Collection* and *Tracking* databases that are used with XP and XP0. You can either host MongoDB with a PaaS service such as mLab, or you can choose to self-host MongoDB using Azure virtual machines.

  Note

  From Sitecore version 9.0 and later, this is not a pre-requisite.

- .NET Framework 4.6
- PowerShell 4.0
- Microsoft Azure PowerShell® 2.0.1 or later. To check your installed version, open PowerShell and run the following command:

  ```
  Get-Module -ListAvailable -Name Azure -Refresh
  ```

  You can install Microsoft Azure PowerShell® using the Microsoft® Web Platform Installer or Windows PowerShell commands.

- Microsoft SQL Server Data-Tier Application Framework (DacFX) for SQL server 2012 or later.

  Note

  You usually install this framework with Microsoft SQL Server® or Microsoft Visual Studio®. You can also download DAC Fx from the Microsoft Download Center. DacFX is also installed with Microsoft Web Deploy 3.6 using the Web Platform Installer. In some software configurations Sitecore Azure Toolkit might not be able to load a proper version of DacFx automatically.

- An Azure subscription for deployments. You can sign up for a free trial account on the Microsoft Azure website.
- Cloud-hosted storage for Sitecore WebDeploy packages. An Azure Storage account can be used for this purpose.

## Downloading the Sitecore Azure Toolkit

You can download the Sitecore Azure Toolkit from dev.sitecore.net, where you can also find the tools and resources necessary to package and deploy Sitecore solutions to the Azure App Service.

Note

When you download the Sitecore Azure Toolkit, the zip file is marked as *blocked* by Microsoft Windows therefore you must unblock the zip file. In Windows Explorer, right-click the file and click Properties. Then in the Properties dialog, on the General tab, click Unblock.

The following resources are distributed separately:

- Prebuilt WebDeploy packages for Sitecore roles in different environments on the Sitecore Experience Platform download page.
- Azure Resource Manager (ARM) provisioning templates and related parameter files on Github.

## Installing the Sitecore Azure Toolkit

Sitecore Azure Toolkit is a ZIP package that you can unpack into a folder of the hard drive. When unpacked, Sitecore Azure Toolkit package creates the following files and folders:

- *Tools* folder – Microsoft PowerShell commandlets and the necessary supplementary libraries.
  - Sitecore.Cloud.Cmdlets.psm1 – the main module containing the necessary commandlets.
- *Resources/9.0.0* – Version-specific resources for Sitecore 9.0.
  - *CargoPayloads* – role-specific and feature-specific transformation packages. They are also known as Sitecore Cargo Payload packages (SCCPL).
  - *MsDeployXmls* – parameters files for WebDeploy packages.
  - *Configs* – configuration files for packaging tasks.
- Top-level files and a *Copyrights* folder – README and licensing information.

To load the main module into a PowerShell session, run the following command in the folder where the toolkit is unpacked:

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

This command loads commandlets into the PowerShell session. Commandlets provide an interface into the functionality of the Sitecore Azure Toolkit, for example:

- `Start-SitecoreAzurePackaging` – packages a Sitecore solution into WDP(s) so they are ready for deployment.
- `Start-SitecoreAzureDeployment` – initiates a Sitecore deployment onto Azure that includes all of the required resource infrastructure.

## Azure Resource Manager templates

ARM templates are located on Github and provide a description of Sitecore environments hosted on Microsoft Azure App Service. The templates include definitions of the following resources:

- Hosting plans for Sitecore instances.
- Microsoft Azure SQL databases for content and reporting.
- Redis Cache service for session state.

- Microsoft Application Insights® for diagnostics.
- Microsoft Azure Search for content search and contact segmentation index.

The templates are compatible with the WebDeploy packages available on dev.sitecore.net for the corresponding Sitecore version or those that are produced by the Sitecore Azure Toolkit. The structure and organization of the templates are nested. The main template references 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.

## Web Deployment Packages

Prebuilt WDPs are located on the Sitecore Experience Platform download page. The WDP(s) come in a single .zip file and are grouped per version and per topology. After downloading the WDPs you must extract them from the .zip file before the Sitecore Azure Toolkit can use them.

Each role of the topology sizes intended for production use is contained within its own WDP. This means, for example that there will be a WDP for the *Content Delivery* role, a WDP for the *Content Management* role, and so on. The only exception to this is for single topologies that are designed for development and testing purposes. For these topologies you can combine multiple roles into a single WDP for deployment on a single shared resource.

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

Note

With appropriate configuration changes, the Sitecore Azure Toolkit also enables users to package their own custom Sitecore deployments into WDPs so they can run on Azure. Further details on how this process works are available within the packaging topics.

## Using the Sitecore Azure Toolkit

Use Sitecore Azure Toolkit to perform the following tasks:

- Package a Sitecore solution for the Microsoft Azure App service
- Deploy a new Sitecore environment to the Microsoft Azure App Service

Send feedback about the documentation to docsite@sitecore.net.

# Package a Sitecore solution for the Microsoft Azure App Service

This document describes how to use the Sitecore Azure Toolkit to package a local Sitecore instance for deployment to the Microsoft Azure App Service®.

## Prerequisites

Before you start the packaging process, make sure you have:

- A local Sitecore instance that you use as input to the packaging process. The Sitecore Azure Toolkit supports packaging Sitecore instances that are installed with the installation program or from a ZIP file and use the default folder structure.
- The Microsoft Data Tier Application Framework (DAC Fx). This framework is usually installed with Microsoft SQL Server® or Microsoft Visual Studio®. You can also download DAC Fx from the Microsoft Download Center.
- The Sitecore Azure Toolkit must be installed according to the instructions described in Getting Started with Sitecore Azure Toolkit.

## Invoke the packaging command

To start the packaging process, import the Sitecore Azure Toolkit PowerShell module and invoke the `Start-SitecoreAzurePackaging` commandlet.

Note

The following commands must be carried out in Microsoft Windows PowerShell® in the folder where the Sitecore Azure Toolkit is installed.

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

```
> Start-SitecoreAzurePackaging [-sitecorePath] <String> [-destinationFolderPath] <String> [-cargoPayloadFolderPath] <String> [-commonC
```

The `Start-SitecoreAzurePackaging` commandlet has the following parameters:

| Parameter | Description |
| --- | --- |
| `sitecorePath` | The path to either a Sitecore instance folder of the Sitecore installation to be packaged or a packaged Sitecore instance folder in a zip file.<br><br>An example of a path to a website folder,<br><br>`C:\inetpub\wwwroot\simple161221` |
| `destinationFolderPath` | The folder where the generated packages are to be stored. |
| `cargoPayloadFolderPath` | The path to the folder containing the Sitecore version-specific role and feature transformation files.<br><br>For example, for Sitecore 8.2 Update-1, this path would go to the subfolder: *resources\8.2.1\cargopayloads* in the Sitecore Azure Toolkit installation. |
| `commonConfigPath` | The path to the file that contains a list of the transformations applied to all of the roles. |

| | |
|---|---|
| | For example, for Sitecore 8.2 Update-1, this file is `common.packaging.config.json` and is located in the Sitecore Azure Toolkit installation, in the *resources\8.2.1\configs* subfolder. |
| `skuConfigPath` | The path to the file that contains the lists of role-specific transformations for the selected Sitecore version and the Sitecore configuration on the Microsoft App Service.<br><br>For example, for Sitecore 8.2 Update-1, these files are located in the Sitecore Azure Toolkit installation, in the resources\8.2.1\configs subfolder and have the following names:<br><br>• `xm1.packaging.config.json` – for an XM1 configuration.<br>• `xp1.packaging.config.json` – for an XP1 configuration.<br>• `xp0.packaging.config.json` – for an XP0 configuration. |
| `archiveAndParameterXmlPath` | The path to the WebDeploy Sitecore version-specific archive manifest and parameter declaration files.<br><br>For example, for Sitecore 8.2 Update-1, these files are located in the Sitecore Azure Toolkit installation, in the *resources\8.2.1\msdeployxmls* subfolder. |
| `fileVersion` | An optional parameter that allows you to embed a version marker in the generated WebDeploy packages. The marker is stored in the `Version.txt` file inside the generated packages. |

When performed, the Start-SitecoreAzurePackaging commandlet produces a number of WebDeploy packages in the destination folder. All packages follow the same name pattern, which includes the role name, for example, `Sitecore 8.2 rev. 161115_cm.wdp.zip`.

Note

Sometimes the commandlet also generates an intermediate WDP package with no role-specific suffix in its name. Ignore this package during deployment.

## Packaging Example

The following code sample shows how to use the Start-SitecoreAzurePackaging commandlet to prepare WDP packages for XP1 configuration. The sample assumes the following:

- Sitecore is installed locally in `C:\inetpub\example`, and the website folder is `C:\inetpub\example\Website`.
- Sitecore Azure Toolkit is installed in `C:\Tools\SitecoreAzureToolkit.`
- Generated packages are stored in `C:\Workspace\WDPs.`

```
$SAT="C:\Tools\SitecoreAzureToolkit"

# Import commandlets

Import-Module "$SAT\tools\Sitecore.Cloud.Cmdlets.psm1"

# Set the parameter variables

$SKU="xp1"

$Version="8.2.1"

$Resources="$SAT\resources\$Version"

$Website="C:\inetpub\example\Website"

$Output="C:\Workspace\WDPs"

# Create the output folder

mkdir $Output

# Start the packaging process

Start-SitecoreAzurePackaging -sitecorePath "$Website" -destinationFolderPath $Output -cargoPayloadFolderPath "$Resources\cargopay
```