Kubernetes を使用した Sitecore 10.1.0 本番環境のデプロイへのインストール ガイド

Sitecore Experience Platform 10.1.0 を Kubernetes インストールにデプロイする方法

目次

1. Azure Kubernetes Service の概要

Sitecore Experience Platform (SXP) は、本番環境をデプロイするためのデフォルトのオーケストレーターとして Kubernetes (K8s) を使用します。

このガイドでは、SXP のコンテナーを Azure Kubernetes Service (AKS) にデプロイする方法について説明します。AKS は、Microsoft のクラウドでホストされる Kubernetes サービスであり、Blob Storage や負荷分散などの Azure 固有の機能を利用する追加機能を備えています。

リファレンスとして、Sitecore ソフトウェア コンテナー間で最低限必要な設定パラメーターをマップするために使用する SXP Kubernetes 仕様ファイルが、SXP コンテナー パッケージで提供されます。これらの仕様を拡張して、独自の要件をサポートすることができます。本番環境が組織で定められた安定性とセキュリティの基準を満たすようにすることは、ユーザーの責任です。

SXP の Kubernetes 仕様ファイルは、可能な限り Azure 固有の依存関係を回避するように設計されています。

1.1. Kubernetes でサポートされている Sitecore トポロジ

Sitecore XP は、Kubernetes で使用する次のトポロジをサポートしています。

- XM サーバー (XM スケーリング)
- XP サーバー (XP スケーリング)

1.1.1. XM サーバー (XM スケーリング)

Kubernetes トポロジ用の Sitecore Experience Manager サーバーは *XM1* とも呼ばれ、本番環境と非本番環境の両方での使用に適しています。

XM1 トポロジは、次の Sitecore ロールをサポートしています。

ロールの種類	Sitecore ロール
本番環境	Content Management
	Content Delivery
	Sitecore Identity Server
	MSSQL Data Initialization Job
	SolrCloud Data Initialization Job
非本番環境	Microsoft SQL Server
	SolrCloud
	RedisLabs Redis サーバー

1.1.2. XP サーバー (XP スケーリング)

Kubernetes トポロジ用の Sitecore Experience Platform サーバーは *XP1* とも呼ばれ、本番環境と非本番環境の両方での使用に適しています。

たとえばテスト目的などで非本番環境で XP1 トポロジをデプロイする場合、本番環境の設定をシミュレーションするにはかなりの量のリソースが必要になることに注意してください。

XP1 トポロジは、次の Sitecore ロールをサポートしています。

ロールの種類	Sitecore ロール
本番環境	Content Management
	Content Delivery
	Sitecore Identity Server
	XDB Processing
	XDB Collection service
	XDB Search service
	Marketing Automation Engine
	Marketing Automation Reporting
	XDB Reference Data service
	Sitecore Cortex Processing service
	Sitecore Cortex Reporting service
	XDB Search Worker
	Sitecore Cortex Processing
	MSSQL Data Initialization Job
	SolrCloud Data Initialization Job
非本番環境	Microsoft SQL Server
	SolrCloud
	RedisLabs Redis サーバー

1.2. Azure Kubernetes Service での Sitecore の要件

Azure Kubernetes Service 環境に Sitecore コンテナーをデプロイする前に環境が満たさなければならない要件が多くあります。これらには次のものが含まれます。

- Sitecore コンテナー デプロイ パッケージ
- クライアントのソフトウェアの要件
- Kubernetes クラスターのソフトウェアの要件
- Kubernetes クラスターのハードウェアの要件
- 必要な外部データ サービス
- Azure Kubernetes Service の要件
- イングレス コントローラーの要件

1.2.1. Sitecore コンテナー デプロイ パッケージ

Sitecore XP 10.1.0 コンテナー デプロイ パッケージには、Sitecore ソフトウェア クラスター ソリューションをデプロイするために使用する Kubernetes 仕様ファイルが含まれています。

Sitecore コンテナー デプロイ パッケージでサポートされているトポロジの詳細については、「Kubernetes でサポートされている Sitecore トポロジ」を参照してください。

1.2.2. クライアントのソフトウェアの要件

Kubernetes に Sitecore Experience Platform をインストールするには、次のソフトウェアがインストールされている必要があります。

- 次のいずれかのオペレーティング システム:
  - Windows 10 1809 以降
  - Windows Server 1809 以降
- Kubectl 1.16x 以降。プレビュー版以外の最新の安定したバージョンを使用します。
  - サポートされている場所のリストを取得するには、次のコマンドを実行します。

    ```
    az account list-locations
    ```

  - 目的の場所で最新の安定したバージョンを取得するには、次のコマンドを実行します。

    ```
    az aks get-versions --location <location> --output table
    ```

- Helm 3.0.x 以降。これはイングレス コントローラーのデプロイに必要です。
  `choco` を使用して Helm をインストールすることをお勧めします。以下に例を示します。

  ```
  choco install kubernetes-helm
  ```

- AKS に Sitecore 仕様ファイルをデプロイするには、Azure CLI 2.8.0 以降が必要です。

次のソフトウェア パッケージもダウンロードする必要があります。

- Sitecore SXP 10.1.0 コンテナー デプロイ パッケージ

1.2.3. Kubernetes クラスターのソフトウェアの要件

- Kubernetes 1.16.x 以降
  スタートアップ プローブを有効にするには、Kubernetes 1.18.x 以降をお勧めします。
- Windows Server 2019 バージョン 1809

1.2.4. Kubernetes クラスターのハードウェアの要件

Windows Server 2019 クラスターには、次のハードウェア要件があります。

- RAM
  起動時の Kubernetes クラスターごとに 16 GB 以上の RAM を推奨します。
  運用の要件は、使用する Azure サービスによって異なります。
- CPU
  起動時の Kubernetes ノードごとにクアッド コア以上を推奨します。

運用の要件は、使用する Azure サービスによって異なります。

- ディスク
  Docker コンテナーのダウンロード時と実行時の最適なパフォーマンスのためにプレミアム SSD ディスクを推奨します。
  運用で必要な容量は、使用するサービスによって異なります。

1.2.5. 必要な外部データ サービス

本番環境では、外部データ サービスを Sitecore XP クラスターの外部でホストする必要があります。

開発とテストに必要な時間を削減するために、Kubernetes の外部サービス サンプルのデプロイは、本番環境以外での使用のみを目的として提供されています。

次のサービスは必須です。

- Microsoft SQL Server
  - Microsoft SQL Server 2017 または 2019
    または
  - SQL Azure Elastic Database Pool
- Apache Solr Cloud 8.4.0
- RedisLabs Redis 5.0 以降

1.2.6. Azure Kubernetes Service の要件

Sitecore を Azure Kubernetes Service (AKS) にデプロイするには、AKS インストールの次の要件を満たす必要があります。

- 最新の安定したリリースの Kubernetes バージョン 1.16.x 以降で設定された AKS クラスター

  注記
  スタートアップ プローブが Sitecore ソフトウェアのコンテナーが正常に起動したかどうかを確認するには、Kubernetes バージョン 1.18.x 以降が必要です。

- 1 つの Windows Server 2019 バージョン 1809 OS ノード。
- 非本番環境やテストの場合、Windows ノードおよび Linux ノードで推奨される最小の VM のサイズは Standard_D4_v3 です。
- 本番環境の場合、VM のサイズとノードの数は個々の要件によって異なります。

AKS でサポートされている最新バージョンの Kubernetes を取得するには、次の Azure CLI コマンドを実行します。

```
az aks get-versions --location eastus --output table
```

注記
コマンドの `eastus` を調べたい地域に置き換えます。

1.2.7. イングレス コントローラーの要件

Kubernetes のイングレス コントローラーは、Sitecore Kubernetes 仕様ファイルをデプロイするために必要です。

Sitecore Kubernetes 仕様ファイルは、Kubernetes でサポートされているほとんどのイングレス コントローラーで動作しますが、すべてのイングレス コントローラーで同じように動作するわけではありません。詳細については、本番環境でのイングレス コントローラーの設定に関するサード パーティのドキュメントを参照してください。

クライアントの IP アドレスの追跡とパーソナライゼーションをサポートするには、クライアントの発信元 IP アドレスを *X-Forwarded-For* HTTP ヘッダーに保存するようにイングレス コントローラーを設定する必要があります。

IP アドレスの追跡とパーソナライゼーションをすべて有効にするには、Sitecore ソフトウェアの設定にいくつかの追加の変更を加える必要があります。

イングレス コントローラーと Sitecore ソフトウェアの設定の詳細については、「*イングレス コントローラー サービス*」セクションを参照してください。

1.3. Azure Kubernetes Service へのデプロイの準備

Azure Kubernetes Service (AKS) への Sitecore XP コンテナーのデプロイ プロセスを開始する前に、理解しておく必要がある概念と手順がいくつかあります。

これらの概念と手順は次のとおりです。

- Sitecore Kubernetes 仕様ファイルの理解
- Sitecore コンテナー レジストリへのアクセス
- デプロイ用の Kubernetes 仕様ファイルを準備する
- Kubernetes シークレットのデプロイ
- Sitecore ライセンス ファイルを圧縮する
- Identity Server のトークン署名証明書を生成する
- TLS/HTTPS 証明書を生成する
- 外部サービスに非本番環境コンテナー イメージを使用する
- 外部データ サービスのホスティング
- 送信キュー永続ストレージの使用
- イングレス コントローラー サービスの設定
- プライベート コンテナー レジストリの使用

1.3.1. Sitecore Kubernetes 仕様ファイルの理解

Sitecore では、コンテナーを Kubernetes クラスターにデプロイするために使用する Kubernetes 仕様ファイル (`.yaml`) が提供されています。

リモート クライアントの Kubectl (Kube control) を使用して Kubernetes クラスターを設定し、必要な設定の状態を指定します。

Kubectl は、Azure CLI の一部として、またスタンドアロンとして使用できます。

1.3.2. Sitecore コンテナー レジストリへのアクセス

Sitecore XP のコンテナー イメージは、パブリックの Docker コンテナー レジストリでホストされ、認証なしで利用できます。

このパブリック レジストリは、Sitecore Kubernetes 仕様ファイルで使用されるデフォルトのレジストリです。

Sitecore ソフトウェアのコンテナー イメージを起動するには、有効な Sitecore ライセンス ファイルが必要です。

Sitecore コンテナー レジストリは Docker を使用してアクセスするもので、`scr.sitecore.com` でホストされており、署名されたイメージをプルできる Docker コンテンツ トラスト モデルをサポートしています。

1.3.3. デプロイ用の Kubernetes 仕様ファイルを準備する

Sitecore Kubernetes 仕様ファイルをデプロイするには、Kubernetes Kubectl CLI を使用する必要があります。

デプロイを準備するには、次の手順に従います。

1. Sitecore のダウンロード ページから Sitecore Experience Platform 10.1.0 コンテナー デプロイ パッケージをダウンロードし、それをインストールの一時フォルダーに展開します。
2. `k8s\<version>\<topology>` フォルダーを見つけます。`<version>` は Windows のバージョンで、`<topology>` は使用する Sitecore トポロジです。たとえば、`k8s\ltsc2019\xp1` フォルダーです。

   > **注記**
   > Sitecore は現在、Kubernetes 上の Sitecore XP Server (XP1) および Sitecore XM Server (XM1) のトポロジをサポートしています。
   >
   > サポートされているトポロジの詳細については、「Docker でサポートされている Sitecore トポロジ」を参照してください。

3. 仕様を確認するために、フォルダーの内容と Kubernetes 仕様ファイル (`.yaml`) を調査します。

1.3.4. Kubernetes シークレットのデプロイ

Sitecore Kubernetes のデプロイでは、シークレットを使用して、クラスターのコンテナーが使用する文字列を安全に保存します。

シークレットは、データベースのユーザー名、パスワード、TLS 証明書を保存するために使用されます。シークレットはテキスト ファイルと証明書ファイル (`tls.crt`、`tls.key`) に設定され、`./secrets/` フォルダーにある各トポロジの Kubernetes 仕様ファイルに保存されます。

Sitecore のコンテナーをデプロイする前に、Kubernetes クラスターにシークレットをデプロイする必要があります。

Kubernetes クラスターに追加のリソースをデプロイする前に、シークレットのテキスト ファイル (`.txt`、`crt`、.key) を必要な値で更新する必要があります。

シークレットの一覧と個々のシークレットの詳細については、「Kubernetes シークレット リスト」を参照してください。

Sitecore Experience Platform コンテナー パッケージの `k8s\<version>\<topology>\secrets` フォルダーには、1 つのコマンドですべてのシークレットの名前と値をデプロイする Kubectl `kustomization.yaml` ファイルがあります。

1.3.5. Sitecore ライセンス ファイルを圧縮する

Sitecore ライセンス ファイルは通常、エンコードされた文字列の形式で環境変数としてコンテナー インスタンスに渡されます。Sitecore ライセンス ファイルは非常に大きいです。そのため、すべての環境変数に対して Windows が許可する最大サイズを超えないようにファイルを圧縮し、*Base64* でエンコードする必要があります。

> **注記**
> 付録「Sitecore ライセンス ファイルのエンコードおよび圧縮」には、ライセンス ファイルを Base64 で圧縮した文字列に変換して Kubernetes シークレットで使用するためのサンプルの PowerShell スクリプトが含まれています。

ライセンス ファイルを圧縮してエンコードしたら、文字列値をライセンス シークレット テキスト ファイル `sitecore-license.txt` にコピーします。

一部の Sitecore ライセンス ファイルは非常に大きいため、圧縮してもコンテナーと互換性がありません。これは通常、ライセンス ファイルに追加の埋め込み HTML が含まれている場合に発生します。

回避策として、ホストからコンテナー内の `c:\inetpub\wwwroot\app_data\license.xml` ファイルにライセンス ファイルをコンテナー ボリュームとしてマウントできます。詳細と設定例については、Sitecore コンテナー開発のドキュメントを参照してください。

1.3.6. Identity Server のトークン署名証明書を生成する

Sitecore Identity Server には、サーバーとクライアント間で渡されるトークンに署名するための秘密鍵証明書が必要です。この証明書を生成して、それを Base64 で文字列形式にエンコードし、シークレットとして Kubernetes クラスターに保存する必要があります。

付録「Identity Server のトークン署名証明書を作成する」には、自己署名証明書を生成し、シークレットとして使用する文字列を作成するサンプル スクリプトが含まれています。

このサンプル スクリプトは、証明書を作成し、証明書のパスワードを `sitecore-identitycertificatepassword.txt` Kubernetes シークレット テキスト ファイルにコピーします。

Identity Server の証明書を、環境変数として渡すのではなく、ファイルシステムにマウントできます。トークン署名証明書をボリュームとしてマウントする方法の詳細については、「GitHub の Sitecore Docker デモ リポジトリ」を参照してください。

1.3.7. TLS/HTTPS 証明書を生成する

最新のブラウザーの要件を満たし、デフォルトで安全な環境を提供するには、Sitecore のコンテナーをデプロイする前に TLS (Transport Layer Security) 証明書を生成する必要があります。これにより、ブラウザーと Kubernetes のイングレス コントローラー間で安全な通信を行うことができます。

Sitecore XP で使用される Kubernetes のデフォルトのイングレス コントローラーは、NGINX Ingress Controller です。NGINX イングレス コントローラーは、ブラウザーから送信された TLS 接続を終了し、クラスター内の個々の XP コンテナーへのネットワーク トラフィックをプロキシするために使用されます。詳細については、Kubernetes のドキュメントの「イングレスの TLS 設定」と NGINX の「TLS ユーザー ガイド」を参照してください。

HTTPS プロトコルは、Content Management ロールと Identity Server ロールで使用される安全なブラウザーの Cookie をサポートするために必要です。HTTPS は Content Delivery ロールでデフォルトで有効になっていますが、お使いの特定のユース ケースで必要ない場合は無効にできます。

付録「TLS/HTTPS 証明書の作成」には、必要な証明書を生成するサンプル スクリプトが含まれています。

自己署名ルート認証局証明書とホストごとの TLS/SSL 証明書が生成されたら、すべてのクライアントの信頼されたルート認証局ストアにルート認証局証明書をインストールする必要があります。付録「TLS/HTTPS 証明書の作成」のサンプル スクリプトでは、*mkcert* ツールを使用して自己署名ルート認証局証明書を自動的に作成し、それを正しい証明書ストアにインストールします。

1.3.8. 外部サービスに非本番環境コンテナー イメージを使用する

Sitecore Experience Platform (SXP) を本番環境以外で使用するために Kubernetes クラスターにデプロイするのにかかる時間を最小限に抑えるために、Sitecore では外部の Microsoft SQL Server、Solr、RedisLabs Redis の各サービスにオープン ソースのコンテナー イメージを使用します。

`k8s\<version>\<topology>\external` フォルダーには、SXP コンテナー パッケージの外部サービス コンテナー用の yaml 仕様ファイルがあります。

> **警告**
> これらのオープン ソースのコンテナー イメージは、*本番環境以外*でのみ使用します。Sitecore ではサポートしていません。

1.3.9. 外部データ サービスのホスティング

本番環境では、必要な Sitecore XP 外部サービスを Kubernetes クラスターの外部でホストする必要があります。

Sitecore の本番環境の Kubernetes のサポートには、Microsoft SQL Server、SolrCloud、RedisLabs Redis の外部ホスト サービスが必要です。

Sitecore XP を Kubernetes にデプロイする前に、本番環境で使用するために外部サービスをデプロイして設定する必要があります。

必要なデータベース、検索スキーマ、必要なデータをデプロイするために、Sitecore コンテナー デプロイ パッケージには Microsoft SQL Server と SolrCloud のための Kubernetes の *データ初期化*ジョブが含まれています。

> **注記**
> RedisLabs Redis 外部サービスでは、初期化は必要ありません。必要なキャッシュ データベースは、最初の使用時に作成されます。

データ初期化ジョブは、Sitecore ソフトウェアのコンテナーをデプロイする前に完了する必要があります。

データの初期化の詳細については、「Sitecore XP の Azure Kubernetes Service へのデプロイ」の「*データ初期化ジョブのデプロイ*」セクションを参照してください。

アプリケーション データベース ユーザー資格情報

Sitecore Kubernetes 仕様ファイルは、アプリケーション データベース ユーザーを使用してデータベースに接続するコンテナーをデプロイするように設計されています。MSSQL Data Initialization Job は、Sitecore データベースのシークレットで提供される資格情報を使用して、各データベースのアプリケーション データベース ユーザーを作成します。

アプリケーション データベースのユーザー名 Secrets にはデフォルト値があります。それらすべてにパスワードを指定する必要があります。

たとえば、Sitecore Master database への接続に使用するアプリケーション データベース ユーザー資格情報を `sitecore-master-database-username.txt` および `sitecore-master-database-password.txt` シークレット ファイルに指定します。

アプリケーション データベース ユーザー資格情報のシークレットの一覧については、付録「Kubernetes シークレット リスト」を参照してください。

1.3.10. 送信キュー永続ストレージの使用

Kubernetes の永続ボリューム クレーム機能を使用して、複数のコンテンツ配信インスタンス間で Sitecore 送信キューを永続化できます。

永続ボリュームは、次の 2 つのいずれかを参照できます。

- Windows ノード上のすべてのセッションを保存する永続ボリューム。
- すべてのセッションを保存するために Azure Storage アカウントをデプロイするストレージ クラス。このアプローチは、Azure Kubernetes Services のベスト プラクティスです。

1.3.11. イングレス コントローラー サービスの設定

Sitecore Kubernetes 仕様ファイルをデプロイするには、Kubernetes のイングレス コントローラーが必要です。イングレス コントローラーは、ブラウザーと Sitecore ソフトウェアのコンテナー間のリバース プロキシとして機能します。

クライアントの IP アドレスの追跡とパーソナライゼーションをサポートするには、クライアントの発信元 IP アドレスを `X-Forwarded-For` HTTP ヘッダーに保存するようにイングレス コントローラーを設定する必要があります。

クライアントの送信元 IP アドレスを保持するには、イングレス コントローラーの `externalTrafficPolicy` 設定を `Local` に設定します。詳細については、「Kubernetes のドキュメント」および「NGINX Ingress Controller の設定」ドキュメントを参照してください。

Sitecore 設定でプロキシ サーバーの使用を明示的に有効にする必要があります。プロキシ サーバーのセットアップの詳細については、Sitecore Experience Platform の「ドキュメント」を参照してください。

Kubernetes で使用するためにこの設定を変更するには、Content Management ロールと Content Delivery ロールの両方に対して、この設定の変更を含む新しいコンテナー イメージを作成することをお勧めします。

設定を変更して新しいコンテナー イメージを構築する方法の詳細については、「GitHub の Sitecore Docker サンプル リポジトリ」を参照してください。

1.3.12. プライベート コンテナー レジストリの使用

認証にはプライベート コンテナー レジストリを使用できます。Kubernetes クラスターがプライベート コンテナー レジストリで認証され、そこからイメージをプルできるようにするには、イメージ プル シークレットを作成する必要があります。

すべての Sitecore ロールの Kubernetes デプロイ仕様では、イメージ プル シークレットの使用をサポートしています。シークレット名は `sitecore-docker-registry` にする必要があります。以下に例を示します。

```
imagePullSecrets:
- name: sitecore-docker-registry
```

プライベート レジストリからプルされたイメージの `.yaml` ファイルのレジストリ パスを変更する必要があります。

2. Sitecore XP の Azure Kubernetes Service へのデプロイ

このセクションでは、Sitecore Experience Platform を Azure Kubernetes Service にデプロイするために実行する必要がある手順について説明します。ここで説明されている順序で手順を実行する必要があります。

2.1. 外部データ サービスのデプロイ

本番環境では、Kubernetes で Sitecore コンテナーをデプロイする前に、必要な外部サービスをデプロイする必要があります。

必要な外部サービスは次のとおりです。

- Microsoft SQL Server *または* SQL Azure Elastic Database Pool
- SolrCloud
- RedisLabs Redis

本番環境では、Kubernetes クラスター内に外部サービスを作成できます。詳細については、セクション「本番環境以外のデプロイでの外部サービスのデプロイ」を参照してください。

2.2. AKS クラスターの作成

Windows Server 2019 ノード プールを使用して新しい Azure Kubernetes Service (AKS) クラスターを作成するには、Azure コマンドライン インターフェイス (Azure CLI) または Azure Portal UI を使用します。AKS クラスターには、1 つ以上のノードを持つ Windows Server 2019 バージョン 1809 ノード プールが 1 つ含まれている必要があります。

Azure CLI を使用して AKS クラスターを作成する詳細については、「Azure AKS のドキュメント」を参照してください。

2.3. kubectl コンテキスト クラスターの設定

Kubectl コンテキスト クラスターを設定するには、次の手順に従います。

1. Azure CLI にログインし、サブスクリプションを設定します。以下に例を示します。

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

2. AKS クラスターで作成された Kubernetes クラスターの資格情報を取得し、ローカルに保存します。以下に例を示します。

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

2.4. イングレス コントローラーをデプロイする

イングレス コントローラーをデプロイするには、次の手順に従います。

1. Windows AMD64 バイナリを使用して Helm をインストールします。また、「Installing Helm Through Package Managers」で説明されている代替方法を使用することもできます。
2. NGINX イングレス コントローラー フィードを Helm に追加します。以下に例を示します。

```
helm repo add stable https://kubernetes.github.io/ingress-nginx
```

3. Helm を使用して NGINX イングレス コントローラーをデプロイします。以下に例を示します。

```
helm install nginx-ingress ingress-nginx/ingress-nginx --set controller.replicaCount=2 --
set controller.nodeSelector."beta\.kubernetes\.io/os"=linux --set
defaultBackend.nodeSelector."beta\.kubernetes\.io/os"=linux --set
controller.admissionWebhooks.patch.nodeSelector."beta\.kubernetes\.io/os"=linux --set-
string controller.config.proxy-body-size=10m
```

> **注記**
> この例では、*proxy-body-size* パラメーターにより、メディア データのアップロードや Sitecore パッケージのインストールなどのペイロードの要求が 10 MB に制限されます。これは、インストールに合わせて調整できます。イングレス設定の詳細については、「NGINX Ingress Controller の設定」を参照してください。

4. 選択したトポロジのルート フォルダーから、次のコマンドを実行します。

```
kubectl apply -k ./ingress-nginx/
```

2.5. シークレットのデプロイ

シークレットをデプロイするには、次の手順に従います。

1. `/secrets` および `overrides\<topology>\secrets` フォルダー内のすべてのシークレット ファイル (.txt、.crt、.key) が「Kubernetes シークレット リスト」に示す要件に従って更新されていることを確認します。

2. 選択したトポロジのフォルダーから、次のコマンドを実行します。

```
kubectl apply -k ./secrets/
```

2.6. 本番環境以外のデプロイでの外部サービスのデプロイ

本番環境では、Kubernetes クラスター外に外部サービスをデプロイする必要があります。ただし、本番環境以外では、Kubernetes クラスター内にサービスをデプロイできます。

本番環境以外のデプロイで外部サービスをデプロイするには、次の手順に従います。

1. 選択したトポロジのフォルダーから、次のコマンドを実行します。

```
kubectl apply -k ./external/
```

2. ポッドのステータスを確認するには、次のコマンドを実行します。

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

3. すべてのポッドのステータスが *実行中/OK* になるまで待つには、次のコマンドを実行します。

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

2.7. データ初期化ジョブのデプロイ

データ初期化ジョブをデプロイするには、次の手順に従います。

1. SolrCloud プロバイダーとして SearchStax を使用する場合は、「Solr の SearchStax プロバイダーのデータを初期化する」の手順に従います。Searchstax を使用しない場合は、選択したトポロジのフォルダーに移動し、次のコマンドを実行します。

```
kubectl apply -k ./init/
```

2. ジョブのステータスを確認するには、次のコマンドを実行します。

```
kubectl get jobs -o wide
```

3. すべてのジョブのステータスが完了/OK になるまで待つには、次のコマンドを実行します。

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

2.8. 永続的なボリューム クレームのデプロイ

永続的なボリューム クレームをデプロイするには、次のコマンドを実行します。

```
kubectl apply -f ./volumes/azurefile
```

2.9. Sitecore ポッドのデプロイ

Sitecore ポッドをデプロイするには、次の手順に従います。

1. 選択したトポロジのフォルダーから、次のコマンドを実行します。

   ```
   kubectl apply -k ./
   ```

2. ポッドのステータスを確認するには、次のコマンドを実行します。

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

3. すべてのポッドのステータスが実行中/OK になるまで待つには、次のコマンドを実行します。

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

2.10. ローカル ホスト ファイルの更新

ローカル ホスト ファイルを更新するには、次の手順に従います。

1. CM ロールのイングレス コントローラー サービスの外部 IP アドレスを取得するには、次のコマンドを実行します。

   ```
   kubectl get service nginx-ingress-ingress-nginx-controller
   ```

2. ローカル ホスト ファイルを、イングレス コントローラーで必要な外部 IP アドレスとホスト名で更新します。デフォルトのホスト名は次のとおりです。

   - `cm.globalhost`
   - `cd.globalhost`
   - `Id.globalhost`

2.11. SolrCloud の検索インデックスの設定

コンテナーのデプロイが完了したら、すべての検索インデックスを更新する必要があります。

SolrCloud の検索インデックスを設定するには、次の手順に従います。

1. ブラウザーを開き、https://cm.globalhost/sitecore に移動します。
2. 管理者ユーザーとシークレットとして設定したパスワードを使用して Sitecore にログインします。
3. Sitecore の [コントロール パネル] で [**管理スキーマを生成**] をクリックします。[**スキーマ生成**] ダイアログ ボックスですべてのインデックスを選択して、[**生成**] をクリックします。
4. Sitecore の [コントロール パネル] の [**インデックス**] セクションで、[**インデックス マネージャーを開く**] をクリックします。
5. [**インデックス マネージャー**] ダイアログ ボックスで、再構築するインデックスを選択し、[**再構築**] をクリックします。
6. 検索インデックスが再構築されたら、[**閉じる**] をクリックします。

3. カスタム モジュールのデプロイ

Sitecore コンポーネントの場合、デフォルトでは、プラットフォームに含まれる Solr コレクションと dacpac のみをデプロイします。Sitecore 10.1 以降のバージョンでは、カスタム モジュールをデプロイすることもできます。

デプロイするモジュールにデータベースの更新やカスタムの Solr コレクションが必要な場合は、`mssql-init` や `solr-init` のイメージ上に新しいレイヤーを構築する必要がある場合があります。モジュール アセット イメージは、モジュール dacpacs や Solr コレクション設定ファイルのソースとして使用する必要があります。

3.1. データベース更新のモジュールへの追加

モジュール データベースの更新を追加するには、次の手順に従います。

1. `mssql` イメージ上に新しいレイヤーを作成します。
2. モジュール dacpacs をモジュール アセット イメージから `c:\resources\module_name_data` フォルダーにコピーします。
3. `mssql-init` イメージの Kubernetes 仕様で、カスタム モジュールを DATABASES_TO_DEPLOY 変数に追加します。

3.2. Solr コレクションのモジュールへの追加

Solr コレクション モジュールを追加するには、次の手順に従います。

1. `solr-init` イメージ上に新しいレイヤーを作成します。
2. Solr コレクション モジュール設定ファイルをモジュール アセット イメージから `c:\data` フォルダーに追加します。

   注記
   コレクションには JSON ファイル形式を使用する必要があります。たとえばモジュール名が `sxa` の場合は、Solr コレクション ファイルに `cores-sxa.json` という名前を付けることができます。

3. `solr-init` イメージの Kubernetes 仕様で、カスタム モジュールを SOLR_COLLECTIONS_TO_DEPLOY 変数に追加します。

   注記
   変数の 1 つに複数のモジュールを追加する場合は、モジュール名をコンマで区切る必要があります。例: SOLR_COLLECTIONS_TO_DEPLOY="sxa,jss"

4. 付録

付録には、さまざまなヘルパー関数と参考資料が含まれています。

4.1. Sitecore ライセンス ファイルのエンコードおよび圧縮

Sitecore ライセンス ファイルを圧縮して *Base64* でエンコードする必要があります。この手順では、ライセンス ファイルを処理し、エンコードされたライセンス キーを `.\secrets\sitecore-license.txt` ファイルに保存します。

ライセンス ファイルを圧縮およびエンコードするには、次の手順に従います。

1. PowerShell コマンド ファイルを作成し、その中に次の関数を入力します。

```
function ConvertTo-CompressedBase64String {
  [CmdletBinding()]
  Param (
    [Parameter(Mandatory)]
    [ValidateScript( {
          if (-Not ($_ | Test-Path) ) {
            throw "The file or folder $_ does not exist"
          }
          if (-Not ($_ | Test-Path -PathType Leaf) ) {
            throw "The Path argument must be a file. Folder paths are not allowed."
          }
          return $true
        })]
    [string] $Path
  )
  $fileBytes = [System.IO.File]::ReadAllBytes($Path)
  [System.IO.MemoryStream] $memoryStream = New-Object System.IO.MemoryStream
  $gzipStream = New-Object System.IO.Compression.GzipStream $memoryStream, ([IO.Compression.CompressionMode]::Compress)
  $gzipStream.Write($fileBytes, 0, $fileBytes.Length)
  $gzipStream.Close()
  $memoryStream.Close()
  $compressedFileBytes = $memoryStream.ToArray()
  $encodedCompressedFileData = [Convert]::ToBase64String($compressedFileBytes)
  $gzipStream.Dispose()
  $memoryStream.Dispose()
  return $encodedCompressedFileData
}
```

2. `Path` パラメーターでライセンス ファイルへのパスを指定して、PowerShell コマンド ファイルを実行します。以下に例を示します。

```
ConvertTo-CompressedBase64String -Path .\license.xml | Out-File -Encoding ascii -NoNewline
-Confirm -FilePath .\secrets\sitecore-license.txt
```

4.2. Identity Server のトークン署名証明書を作成する

次の PowerShell スクリプトを使用して、Identity Server のトークン署名証明書を作成します。

```
$certificatePassword = "Test123!"

$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 $certificatePassword -Force -AsPlainText)

[System.Convert]::ToBase64String([System.IO.File]::ReadAllBytes((Get-
Item .\SitecoreIdentityTokenSigning.pfx))) | Out-File -Encoding ascii -NoNewline -Confirm -
FilePath .\secrets\sitecore-identitycertificate.txt
```

4.3. TLS/HTTPS 証明書の作成

NGINX イングレス コントローラーに必要な TLS/SSL 証明書を生成するには、次の手順に従います。

1. 管理者権限で Windows コマンド プロンプトを開きます。
2. 使用しているトポロジの Kubernetes 仕様ファイルを含むフォルダーに移動します。
3. 次のコマンドを 1 つずつ実行します。

```
IF NOT EXIST mkcert.exe powershell Invoke-WebRequest https://github.com/FiloSottile/mkcert/
releases/download/v1.4.1/mkcert-v1.4.1-windows-amd64.exe -UseBasicParsing -OutFile
mkcert.exe

mkcert -install

del /Q /S *.crt

del /Q /S *.key

mkcert -cert-file secrets\tls\global-cm\tls.crt -key-file secrets\tls\global-cm\tls.key
"cm.globalhost"

mkcert -cert-file secrets\tls\global-cd\tls.crt -key-file secrets\tls\global-cd\tls.key
"cd.globalhost"

mkcert -cert-file secrets\tls\global-id\tls.crt -key-file secrets\tls\global-id\tls.key
"id.globalhost"
```

> **注記**
> `mkcert` ユーティリティを初めて実行すると、生成された自己署名ルート認証局をインストールするように求められます。

4.4. Solr の SearchStax プロバイダーのデータを初期化する

SolrCloud プロバイダーとして SearchStax を使用する場合、データを初期化するには、以下の手順に従います。

1. `.\overlays\init\SearchStax` フォルダーで次のファイルを見つけて、SearchStax 固有のシークレットを入力します。

ファイル名	シークレットの説明	トポロジ
`sitecore-searchstax-account-name.txt`	SearchStax アカウント名	XM1、XP1
`sitecore-searchstax-apikey.txt`	SearchStax API キー	XM1、XP1
`sitecore-searchstax-deployment-uid.txt`	SearchStax デプロイ UID	XM1、XP1

2. 次のコマンドを実行します。

```
kubectl apply -k .\overlays\init\SearchStax
```

4.5. Kubernetes シークレット リスト

次の表で Kubernetes Secrets ファイルを説明します。

名前	説明	トポロジ	デフォルト値
`sitecore-license.txt`	GZIP で圧縮され、Base64 でエンコードされた文字列に変換されたライセンス ファイルの内容 (「Sitecore ライセンス ファイルのエンコードおよび圧縮」を参照してください)	XM1、XP1	
`sitecore-adminpassword.txt`	Sitecore アプリケーション管理者のパスワード	XM1、XP1	
`sitecore-telerikencryptionkey.txt`	Telerik の Web コントロールによって使用される対称鍵 長さ: 64 - 128 文字	XM1、XP1	
`sitecore-identitycertificate.txt`	データの暗号化に使用される Identity Server の証明書。 (「Identity Server のトークン署名証明書を作成する」を参照してください)	XM1、XP1	
`sitecore-identitycertificatepassword.txt`	Identity Server 証明書を開くためのパスワード	XM1、XP1	
`sitecore-identitysecret.txt`	Identity Server とクライアント ロール間の共有シークレット 長さ: 64 文字	XM1、XP1	

名前	説明	トポロジ	デフォルト値
`sitecore-reportingapikey.txt`	長さ: 64 - 128 文字	XP1	
`sitecore-solr-connection-string.txt`	Solr の接続文字列	XM1、XP1	http://solr:8983/solr;solrCloud=true
`sitecore-solr-connection-string-xdb.txt`	Solr-xdb の接続文字列 `sitecore-solr-core-prefix-name.txt` ファイルでカスタム プレフィックスを指定した場合、接続文字列を同じプレフィックスで更新する必要があります。	XP1	http://solr:8983/solr/sitecore_xdb;solrCloud=true
`sitecore-solr-prefix-name.txt`	Solr コア名の共通プレフィックス。 XP1 トポロジを使用してこの値を変更する場合は、`sitecore-solr-connection-string-xdb.txt` ファイルの solr xdb 接続文字列を同じプレフィックスで更新する必要があります。	XM1、XP1	sitecore
`sitecore-databaseusername.txt`	SQL Server 管理者名	XM1、XP1	sa
`sitecore-databaseservername.txt`	MS SQL Server に接続するサーバー名	XM1、XP1	mssql
`sitecore-databasepassword.txt`	MS SQL Server に接続するためのパスワード	XM1、XP1	
`sitecore-database-elastic-pool-name.txt`	データベースのエラスティック プール名。 SQL Server にアクセスできるエラスティック プール リソースの名前を使用します	XM1、XP1	
`sitecore-core-database-username.txt`	MS SQL Server のデータベース名 *_Core の UserName	XM1、XP1	coreuser
`sitecore-core-database-password.txt`	MS SQL Server のデータベース名 *_Core の Password	XM1、XP1	
`sitecore-master-database-username.txt`	MS SQL Server のデータベース名 *_Master の UserName	XM1、XP1	masteruser
`sitecore-master-database-password.txt`	MS SQL Server のデータベース名 *_Master の Password	XM1、XP1	
`sitecore-web-database-username.txt`	MS SQL Server のデータベース名 *_Web の UserName	XM1、XP1	webuser
`sitecore-web-database-password.txt`	MS SQL Server のデータベース名 *_Web の Password	XM1、XP1	
`sitecore-forms-database-username.txt`	MS SQL Server のデータベース名 *_ExperienceForms の UserName	XM1、XP1	formsuser
`sitecore-forms-database-password.txt`	MS SQL Server のデータベース名 *_ExperienceForms の Password	XM1、XP1	
`sitecore-exm-master-database-username.txt`	MS SQL Server のデータベース名 *_EXM.Master の UserName	XP1	exmmasteruser
`sitecore-exm-master-database-password.txt`	MS SQL Server のデータベース名 *_EXM.Master の Password	XP1	

名前	説明	トポロジ	デフォルト値
sitecore-messaging-database-username.txt	MS SQL Server のデータベース名 *_Messaging の UserName	XP1	messaginguser
sitecore-messaging-database-password.txt	MS SQL Server のデータベース名 *_Messaging の Password	XP1	
sitecore-marketing-automation-database-username.txt	MS SQL Server のデータベース名 *_MarketingAutomation の UserName	XP1	mauser
sitecore-marketing-automation-database-password.txt	MS SQL Server のデータベース名 *_MarketingAutomation の Password	XP1	
sitecore-processing-engine-storage-database-username.txt	MS SQL Server のデータベース名 *_ProcessingEngineStorage の UserName	XP1	processingenginestorageuser
sitecore-processing-engine-storage-database-password.txt	MS SQL Server のデータベース名 *_ProcessingEngineStorage の Password	XP1	
sitecore-processing-engine-tasks-database-username.txt	MS SQL Server のデータベース名 *_ProcessingEngineTasks の UserName	XP1	processingenginetasksuser
sitecore-processing-engine-tasks-database-password.txt	MS SQL Server のデータベース名 *_ProcessingEngineTasks の Password	XP1	
sitecore-processing-pools-database-username.txt	MS SQL Server のデータベース名 *_Processing.Pools の UserName	XP1	processingpoolsuser
sitecore-processing-pools-database-password.txt	MS SQL Server のデータベース名 *_Processing.Pools の Password	XP1	
sitecore-processing-tasks-database-username.txt	MS SQL Server のデータベース名 *_Processing.Tasks の UserName	XP1	processingtasksuser
sitecore-processing-tasks-database-password.txt	MS SQL Server のデータベース名 *_Processing.Tasks の Password	XP1	
sitecore-reference-data-database-username.txt	MS SQL Server のデータベース名 *_ReferenceData の UserName	XP1	refdatauser
sitecore-reference-data-database-password.txt	MS SQL Server のデータベース名 *_ReferenceData の Password	XP1	
sitecore-reporting-database-username.txt	MS SQL Server のデータベース名 *_Reporting の UserName	XP1	reportinguser
sitecore-reporting-database-password.txt	MS SQL Server のデータベース名 *_Reporting の Password	XP1	
sitecore-collection-shardmapmanager-database-username.txt	MS SQL Server のデータベース名 *_Xdb.Collection.ShardMapManager の UserName	XP1	shardmapmanageruser
sitecore-collection-shardmapmanager-database-password.txt	MS SQL Server のデータベース名 *_Xdb.Collection.ShardMapManager の Password	XP1	

名前	説明	トポロジ	デフォルト値
`sitecore-media-request-protection-shared-secret.txt`	メディア要求の保護のために使用される Sitecore シークレット。 **重要** 共有シークレットをランダムな文字列に変更する必要があります。デフォルト値を使用しないでください。	XM1、XP1	HQ(NjM(u6_5koVla-cTf4ta8x1h6Sb+ZcUQrULUz-0Afpx0cx-NuMtIoQkpDFmX5

4.6. 一般的な問題

このセクションでは、発生する可能性のある問題の解決策を示します。

4.6.1. 言語ファイルを Web サイトのルート フォルダーにアップロードできません

アップロード操作中に [**ファイルをアップロード**] ダイアログがハングした場合、次のエラーがログ ファイルに書き込まれます。

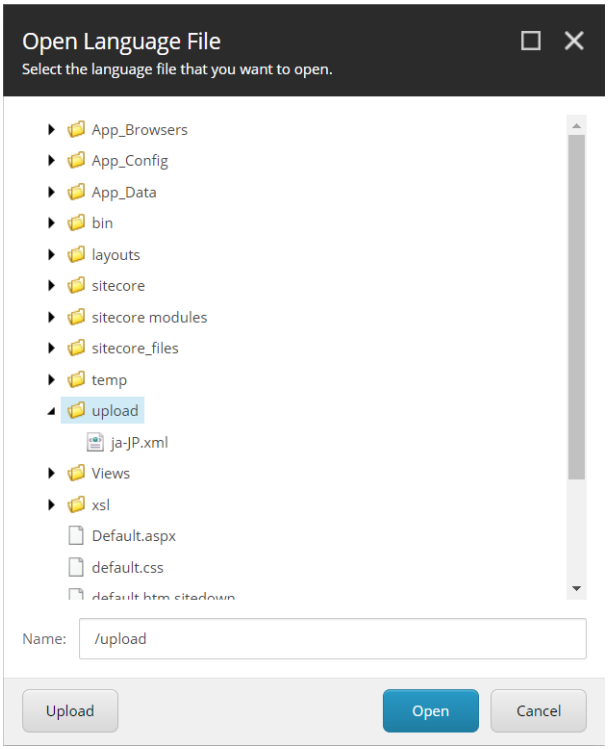
```
ERROR Could not save posted file: ja-JP.xml
Exception: System.UnauthorizedAccessException
Message: Access to the path 'C:\inetpub\wwwroot\ja-JP.xml' is denied.
```

これは、[**言語ファイルをインポートする**] ダイアログでは言語ファイルがデフォルトで `website` のルート フォルダーにアップロードされますが、セキュリティ上の理由からこのフォルダーへの書き込みアクセスが拒否されるために発生します。

この問題を解決するには、書き込みアクセスが有効になっている `upload` フォルダーに言語ファイルをアップロードします。

言語ファイルをアップロードするには、次の手順に従います。

1. [**言語ファイルを開く**] ダイアログで、`upload` フォルダーを選択し、[**アップロード**] をクリックします。



2. [**ファイルをアップロード**] ダイアログで、言語ファイルを `upload` フォルダーにアップロードできるようになります。
3. [**言語ファイルをインポートする**] ダイアログを使用して、言語ファイルを `upload` フォルダーからインポートします。

> **注記**
> 言語の xml ファイルは、メディア ライブラリに保存されます。言語をインポートした後で削除できます。

4.6.2. Sitecore のロールのコンテナーに対して Sitecore のメインのログしか作成されない

Sitecore は LogMonitor ツールを使用してコンテナーのログ ファイルを収集および出力します。デフォルトでは、このツールは次のログ ファイルを監視するように設定されています。

- システム イベントのログ - エラー レベルのエントリ。
- IIS のログ
- Sitecore のメインのログ (`log.*.txt` ファイル) - Sitecore のロール用
- xConnect のログ (`xconnect-log-*.txt` ファイル) - xConnect のロール用

検索、クロール、パブリッシュなどの Sitecore の補助ログは、Sitecore コンテナーでは監視されません。

LogMonitor ツールを再設定して他のロールのログを含めるには、下記の手順に従います。

1. 対応するロール イメージで Dockerfile を作成します。
2. `C:\LogMonitor\LogMonitorConfig.json` ファイルで、`c:\\inetpub\\wwwroot\\App_data\\logs` ディレクトリのソースのフィルターを変更します。以下に例を示します。

```
{
  "LogConfig": {
    "sources": [
      ...
      {
        "type": "File",
        "directory": "c:\\inetpub\\wwwroot\\App_data\\logs",
        "filter": ".*log*.txt",
        "includeSubdirectories": false
      }
    ]
  }
}
```

3. Dockerfile を使用して、ロールの新しい Docker イメージを構築します。

> **注記**
> PowerShell またはコマンド プロンプト ターミナルからコンテナーに接続することにより、コンテナーのファイル システムからすべての Sitecore ログ ファイルを直接表示することもできます。

4.6.3. SSC 要求の際、ヘッダー名にアンダースコアが含まれていると問題が発生する

たとえば、次の GET 要求の結果は `bad request` エラーになります。

- URL: https://cm.globalhost/sitecore/api/ssc/aggregate/content/Items
- ヘッダーを追加する
- キー: sc_apikey
- 値: 作成したアイテムの ID

これは、Nginx がヘッダー名でアンダースコアを許可しないために発生します。

これを解決するには、2 つの方法があります。

- すべてのリクエストのパラメータとして `sc_apikey` を追加します。以下に例を示します。

```
https://cm.globalhost/sitecore/api/ssc/aggregate/content/Items?sc_apikey=336A2458-1E02-412E-AA1B-E84E002264FC
```

- イングレスの設定に ConfigMap を導入することにより、ヘッダー名にアンダースコアを使用できます。詳細については、Kubernetes「ConfigMap」ドキュメントを参照してください。

4.6.4. コンテナーから CM サービスを参照すると接続に失敗する

サービス名の代わりに完全修飾ドメイン名を使用します。たとえば、CM サービスが `default` 名前空間にあり、クラスターのドメイン名が `cluster.local` の場合は、サービスの DNS 名 `cm.default.svc.cluster.local` を使用してください。