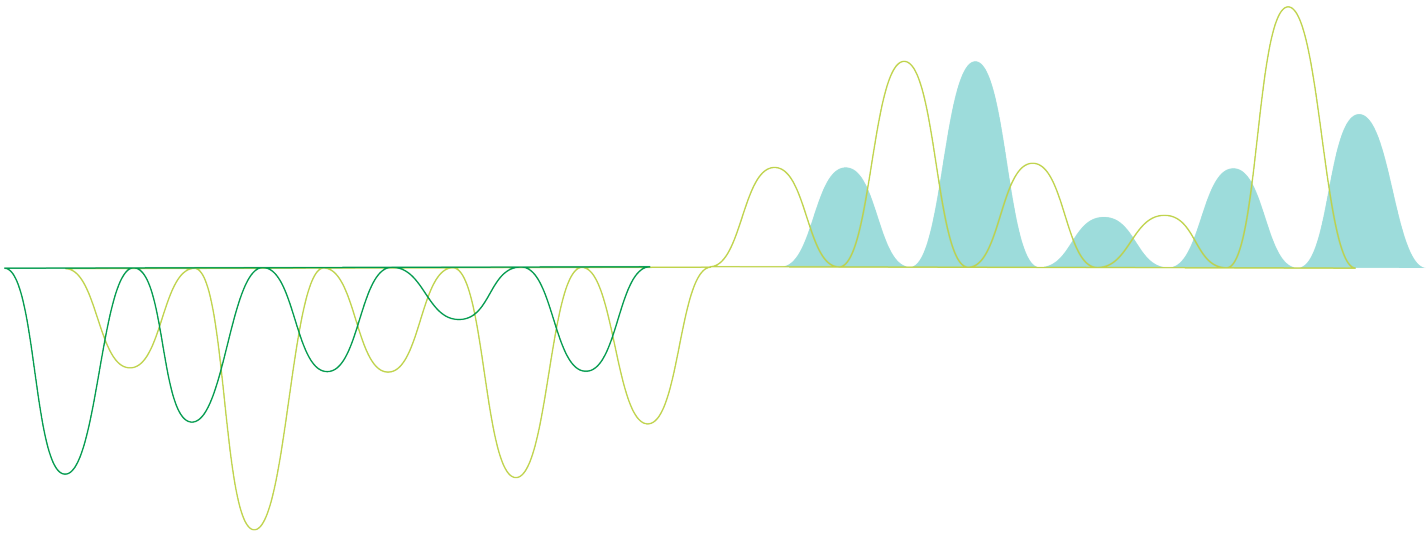
Qlik Sense Enterprise の Kubernetes への展開

Qlik Sense®
November 2019



Contents

1 Qlik Sense Enterprise 展開の計画

Qlik Sense 展開を正しく計画して準備するには、次のことを行います。

Qlik Sense Enterprise について
: Qlik Sense Enterprise の大まかな概要を理解します。

Qlik Sense Enterprise 展開の例
: Qlik Sense Enterprise の展開のさまざまな例を参照してください。

Qlik Sense Enterprise のシステム要件
: Qlik Sense Enterprise のシステム要件を確認します。

Qlik 製品のライセンス
: サイトのライセンス付与のための Qlik Sense によるライセンス キーおよび LEF の使用方法を理解します。
ユーザー アクセス割り当てのための Qlik Sense によるトークンの使用方法を理解します (トークンベースのライセンス付与)。
Qlik Sense ライセンス キーが使用可能であることを確認します。

1.1 Qlik 製品のライセンス

ここでは、Qlik Sense 関連のさまざまな製品で選択可能なライセンス オプションの概要を示します。ライセンス付与により、組織内の Qlik Sense ソフトウェアの使用を管理することができます。

Qlik Sense ライセンス付与オプションの詳細については、Qlik の法的規約、製品規約、および Licensing Service リファレンスガイドを参照してください。

- Qlik Legal Terms (Qlik 法的規約)
- Qlik Product Terms (Qlik 製品規約)
- Qlik Licensing Service Reference Guide「Qlik Licensing Service リファレンスガイド」

製品のアクティベーション

Qlik 製品はすべて LEF (License Enabler File) によって権限が付与され、実行されます。LEF とは、実稼働の際にダウンロードされる中間生成物です。Qlik 製品にライセンスを付与して起動するには、シリアル番号とコントロール ナンバーを使用するか、署名付きライセンス キーを使用します。署名付きライセンス キーは Qlik Sense Enterprise on Cloud Services と Qlik Sense Enterprise on Kubernetes の展開に使用する必要があり、キャパシティベースのライセンスを使用する際にも使います。

ユーザーの割り当て、分析時間消費および製品のアクティベーションには、クラウドベースのライセンス バックエンドにアクセスするために、署名済みのライセンス キーとともにインターネット アクセス (直接またはプロキシ経由) が必要となります。

統一ライセンス

Qlik Sense および QlikView の April 2019 リリース以降、Qlik Sense カスタマーは複数の展開で統一ライセンスを使用できます。統一ライセンスでは、次の展開の組み合わせで同じ署名付きキーを共有します。

- 複数の Qlik Sense Enterprise 展開
- 複数の QlikView Server 展開
- QlikView Server 展開と Qlik Sense Enterprise 展開

同じ署名付きキーを複数の展開に適用すると、同じユーザーおよびアクセス タイプを共有することができます。ユーザーは、同じ Professional または Analyzer のアクセス権割り当てを使用して接続されているすべての展開にアクセスできます。

Qlik Sense Enterprise

Qlik Sense Enterprise は Qlik Sense のサーバー バージョンで、シングル ノードまたはマルチノードに展開することができます。Qlik Sense Enterprise を展開する際にライセンスを付与して起動するには、シリアル番号とコントロール ナンバーを使用するか、署名付きライセンス キーを使用します。お使いの Qlik Sense Enterprise ライセンスは、アクセス タイプまたは トークンのいずれかに基づいています。

アクセス タイプ

アクセス タイプのライセンスは、Professional User と Analyzer User ライセンス (ユーザーベース) および Analyzer Capacity ライセンス (キャパシティベース) です。展開とアクティブ化を行う際に署名付きライセンス キーを使用した場合、サブスクリプション ベースのライセンスとしてこうしたライセンスを組み合わせることも可能です。永続ライセンスを使用している場合は、ユーザーベースのライセンスのみを組み合わせることができます。

署名付きキーのライセンスに変更した後は、古い LEF ベースのライセンス モデルを再び使用することはできません。

トークン

トークンを使用してアクセス パスをユーザーに割り当てて、ユーザーが Qlik Sense にアクセスできるようにします。License Enabler File (LEF) により、さまざまなアクセス パスに割り当てることができるトークンの数が決まります。アクセス パスなしのユーザーはアプリにアクセスできません。

Qlik Sense Token ライセンスでは、トークンを使用してアクセス パスをユーザーに割り当てます。User アクセスおよび Login アクセスを割り当てることができます。

この Token ライセンスは、既存の Qlik Sense Token ライセンスを持つ顧客にのみ付与できます。

コアベースのサイト

Qlik Sense Enterprise コアベースのサイトは、ソフトウェアが動作する CPU コアの数に基づいてライセンスが付与されます。コアとは、プロセッサまたは CPU 内の単一の処理ユニットを意味し、物理か仮想にかかわらず、vCPU または仮想コードを含んでおり、一度に 1 つのソフトウェア スレッドを実行することができます。

Qlik Sense Desktop

Qlik Sense Desktop は、使用するコンピューターにダウンロードできる無償バージョンの Qlik Sense です。www.qlik.com を参照してください。Qlik Sense Desktop をダウンロードするには、Qlik アカウントを作成し、Qlik Sense Desktop ライセンス契約を承諾する必要があります。さらに、ソフトウェアのインストール時には、ソフトウェアのライセンス契約を承諾する必要があります。

Qlik DataMarket

Qlik DataMarket は、Qlik Sense 内で直接アクセス可能な外部ソースから、最新の豊富なデータを提供します。利用可能なデータには、現在と過去の気象および人口統計データや為替レートとともに、ビジネス、経済、および社会に関するデータが含まれます。

Qlik DataMarket のライセンスには、無償版とライセンス版の 2 つのオプションがあります。無償オプションでは、一部のデータセットへのアクセス権が付与されます。ライセンスありオプションでは、プレミアム データ パッケージへのアクセス権が付与されます。ライセンスは、Qlik Sense Enterprise の場合と同じ方法で、所有者情報、シリアル番号、およびコントロール ナンバーを入力して認証します。

> *Qlik Sense Desktop を使用する場合は、Qlik DataMarket の利用規約への同意は必要ありません。資格情報の入力も必要ありません。これは、Qlik Sense Desktop ではプレミアム データ セットが利用できないためです。*

Qlik NPrinting

QlikView のドキュメントまたは Qlik Sense のアプリに接続するように、Qlik NPrinting をインストールして構成できます。ライセンス付与の要件と手順は、Qlik NPrinting を QlikView と Qlik Sense のどちらに接続するのかに応じて異なります。

バージョン 16.0.0.0 以降の Qlik NPrinting には、LEF によってライセンスが付与されます。以前のバージョンの Qlik NPrinting を使用している場合は、Qlik NPrinting バージョン 16.0.0.0 以降へのアップグレードをお勧めします。

> *Qlik Sense トークンは、Qlik NPrinting サービス アカウントの場合には不要です。ただし、Qlik NPrinting サービス アカウント内で頻繁にトラブルシューティングを行う場合は、トークンを Qlik NPrinting サービス アカウントに割り当て、Qlik Sense ハブへのアクセス権が与えられるようにすると便利です。*

Qlik Sense ライセンス

Qlik Sense Enterprise は Qlik Sense のサーバー バージョンで、シングル ノードまたはマルチ ノードに展開することができます。Qlik Sense Enterprise ライセンスは、アクセス タイプまたは トークンのいずれかに基づいています。

Qlik Sense ライセンス付与オプションの詳細については、Qlik の法的規約、製品規約、および Licensing Service リファレンスガイドを参照してください。

- Qlik Legal Terms (Qlik 法的規約)
- Qlik Product Terms (Qlik 製品規約)
- Qlik Licensing Service Reference Guide「Qlik Licensing Service リファレンスガイド」

Qlik Cloud Services または Qlik Sense Enterprise on Kubernetes をセットアップする場合は、Qlik 代理店または Qlik サポートに連絡してセットアップのための有効なライセンスを入力してください。

統一ライセンス

Qlik Sense および QlikView の April 2019 リリース以降、Qlik Sense カスタマーは複数の展開で統一ライセンスを使用できます。統一ライセンスでは、次の展開の組み合わせで同じ署名付きキーを共有します。

- 複数の Qlik Sense Enterprise 展開
- 複数の QlikView Server 展開
- QlikView Server 展開と Qlik Sense Enterprise 展開

同じ署名付きキーを複数の展開に適用すると、同じユーザーおよびアクセス タイプを共有することができます。ユーザーは、同じ Professional または Analyzer のアクセス権割り当てを使用して接続されているすべての展開にアクセスできます。

Qlik Sense Enterprise

Qlik Sense Enterprise 展開は、2 つの異なるモデルである、シリアル番号とコントロール ナンバー、および署名付きライセンス キーを使用してライセンスを付与することができます。License Enabler File (LEF) で、ライセンスの契約条件およびユーザーに割り当て可能なアクセス タイプを定義します。お使いの Qlik Sense Enterprise ライセンスは、アクセス タイプまたは トークンのいずれかに基づいています。コアベース ライセンスも使用できます。署名付きライセンス キーは Qlik Sense Enterprise on Cloud Services と Qlik Sense Enterprise on Kubernetes の展開に使用する必要があり、キャパシティベースのライセンスを使用する際にも使います。

ユーザーの割り当て、分析時間消費および製品のアクティベーションには、クラウドベースのライセンス バックエンドにアクセスするために、署名済みのライセンス キーとともにインターネット アクセス (直接またはプロキシ経由) が必要となります。

ユーザーベースおよびキャパシティベースのライセンス

ユーザーベースのライセンスは、一意に指定されたユーザーに割り当て可能なアクセスの既定数の割り当てを許可します。Qlik Sense Enterprise では、ユーザーベースのライセンスは、Professional User および Analyzer User ライセンス、または Token ライセンスで割り当てられる User アクセス パスのいずれかです。

キャパシティベースのライセンスは、識別されたユーザーまたは匿名ユーザーが使用可能な Qlik Sense Enterprise へのアクセスの既定時間数の割り当てを許可します。Qlik Sense では、キャパシティベースのライセンスは、Analyzer Capacity アクセス、または Token ライセンスで割り当てられた Login アクセス パスのいずれかです。

アクセス タイプ

アクセス タイプのライセンスは、Professional User とAnalyzer User ライセンス (ユーザーベース) および Analyzer Capacity ライセンス (キャパシティベース) です。展開とアクティブ化を行う際に署名付きライセンス キーを使用した場合、サブスクリプション ベースのライセンスとしてこうしたライセンスを組み合わせることも可能です。永続ライセンスを使用している場合は、ユーザーベースのライセンスのみを組み合わせることができます。

署名付きキーによるライセンスに変更した後は、古いシリアル番号とコントロール ナンバーのライセンス モデルを再び使用することはできません。

Professional User および Analyzer User ラインセンス

Professional User および Analyzer User ライセンスは、Professional および Analyzer アクセス タイプで構成されます。

- 識別されたユーザーに Professional アクセス権 (ユーザーベース) を割り当て、Qlik Sense サイト内のストリームとアプリにアクセスできるようにします。Professional アクセスは、Qlik Sense インストールのすべての機能にアクセスする必要があるユーザーが対象です。Professional アクセス権のあるユーザーは、シートまたはアプリを作成、編集、公開でき、Qlik Sense サイトの管理などの使用可能な機能をすべて使うことができます。
- 識別されたユーザーに割り当てられた Analyzer アクセス権によって、ハブ内のストリームとアプリにアクセスできるようにします。アナライザー アクセスは、他のユーザーによって作成されたシートとアプリを使用するユーザーが対象です。アナライザー アクセス権を持つユーザーは、シートやアプリを作成、編集、または公開することはできませんが、アプリ内のデータに基づいて、ストーリー、ブックマーク、スナップショットを作成できます。また、ユーザーは、ブックマーク、印刷オブジェクト、ストーリー、シートを作成したり、オブジェクトから Excel にデータをエクスポートしたりすることもできます。

Analyzer Capacity ライセンス

Analyzer Capacity ライセンスは、Analyzer Capacity アクセス タイプで構成されます。

- アナライザー キャパシティは消費ベースのライセンス タイプで、使用可能な機能に関してアナライザー アクセスに似ています。ユーザーは、ハブ内のストリームとアプリにアクセスし、他のユーザーによって作成されたシートとアプリを消費できます。Analyzer Capacity アクセスでは、アプリ内のデータに基づいてストーリー、ブックマーク、スナップショットを作成し、公開することができます。シートまたはアプリの作成、編集、または公開はできません。
  アナライザー キャパシティ ライセンスにより、アナライザー時間 (毎月の定義済み分時間 (カレンダー日付)) に対してサブスクライブします。これらの分時間は、ユーザー間で共有され、匿名ユーザーを含むユーザー グループの一員になっている任意のユーザーが消費可能です。消費は 6 分単位で測定されます。新しい 6 分間ごとに、1 単位が消費されます。

アクセスの動的割り当て

アクセスの動的割り当ては Qlik Sense Enterprise on Kubernetes および Qlik Sense Enterprise on Cloud Services の展開で使用でき、Management Console で管理されます。

動的割り当てを有効にすることで、ユーザーへのアクセス権の割り当てを簡素化できます。次の 4 つのオプションから選択します。

- Professional とAnalyzer の両方のアクセス権で有効な動的割り当て:
  可能な場合は Professional アクセス権が割り当てられ、不可の場合は Analyzer アクセス権が割り当てられます。どちらのアクセス権も割り当て不可の場合は、可能であれば Analyzer Capacity が割り当られます。
- Professional アクセス権にのみ有効な動的割り当て:
  可能な場合は Professional アクセス権が割り当てられ、不可の場合は、Analyzer Capacity を割り当て可能であれば Analyzer Capacity が割り当てられます。
- Analyzer アクセス権にのみ有効な動的割り当て:
  可能な場合は Analyzer アクセス権が割り当てられ、不可の場合は、Analyzer Capacity を割り当て可能であれば Analyzer Capacity が割り当てられます。
- Professional とAnalyzer の両方のアクセス権で無効な動的割り当て:
  可能であれば、Analyzer Capacity アクセス権が割り当てられます。

Analyzer アクセス権からProfessional アクセス権にアップグレードすることができますが、Professional から Analyzer にダウングレードすることはできません。

新しいライセンス キーに変更すると、すべての割り当てが削除されます。これは、割り当てがテナントではなくライセンスに関連付けられているからです。ただし、古いライセンス キーの使用を再開すれば、割り当てが復元されます。

トークン

トークンを使用してアクセス パスをユーザーに割り当てて、ユーザーが Qlik Sense にアクセスできるようにします。License Enabler File (LEF) により、さまざまなアクセス パスに割り当てることができるトークンの数が決まります。アクセス パスなしのユーザーはアプリにアクセスできません。

この Token ライセンスは、既存の Qlik Sense Token ライセンスを持つ顧客にのみ付与できます。

トークンを使用して割り当て可能なアクセス パスには、2 つのタイプがあります。

- User アクセス パス (ユーザーベース) - 識別された一意のユーザーに割り当てられ、割り当てられたユーザーは、アプリ、ストリーム、その他のリソースに無制限にアクセスできます。
- Login アクセス パス (キャパシティベース) - 頻度の低いアクセスまたは匿名アクセス用に、一連のパスをグループに割り当てます。限られた期間にフル アクセスを許可します。

トークンを割り当てると、利用可能なトークンの数が減ります。各アクセス タイプは特定数のトークンを使用し、トークンの残数がゼロになるか不足すると、そのアクセス タイプに割り当てられなくなります。トークンの使用数を減らして、異なる方法でトークンを使用することができます。Qlik Sense サイトのトークンの数は、新しいライセンスをアクティブにすることで、増やしたり減らしたりできます。

コアベースのサイト

Qlik Sense Enterprise コアベースのサイトは、ソフトウェアが動作する CPU コアの数に基づいてライセンスが付与されます。このライセンスは License Enabler File (LEF) を使用して管理され、Qlik associative engine およびそのコンポーネントが動作できるコアの最大数を制限します。コアとは、プロセッサまたは CPU 内の単一の処理ユニットを意味し、物理か仮想にかかわらず、vCPU または仮想コードを含んでおり、一度に 1 つのソフトウェア スレッドを実行することができます。

ライセンス認証 ファイル

Qlik Sense ではライセンス モデルに 2 つの選択肢があります。シリアル番号とコントロール ナンバー付きのライセンス キー、および署名付きのライセンス キーです。License Enabler File (LEF) で、ライセンスの契約条件およびユーザーに割り当て可能なアクセス タイプを定義します。

シリアル番号とコントロール ナンバーを使用して Qlik Sense のライセンスを付与する場合、Qlik Management Console (QMC) にシリアル番号とコントロール ナンバーが入力されると、LEF がダウンロードされます。利用可能なネットワーク接続がない場合は、LEF を直接 QMC に貼り付けることもできます。シリアル番号とコントロール ナンバーを使用して認証できるライセンス タイプは 2 つあります。Professional User および Analyzer User ライセンスと、Qlik Token ライセンスです。

署名付きのキーを使用して Qlik Sense のライセンスを付与すると、LEF ファイルが License Backend に格納されます。

Qlik Cloud Services または Qlik Sense Enterprise on Kubernetes をセットアップする場合は、Qlik 代理店または Qlik サポートに連絡してセットアップのための有効なライセンスを入力してください。

Professional User および Analyzer User ラインセンス

Professional User および Analyzer User ラインセンスは、Professional および Analyzer (ユーザーベース) アクセス タイプの既定数の割り当てを許可します。LEFファイルによって、アクセス タイプの割り当てが指定されます。

Analyzer Capacity ライセンス (キャパシティベース) は、署名付きのキーを使用してのみ、付与できます。同じ Qlik Sense Enterprise インストールで Professional、Analyzer、および Analyzer Capacity のアクセス タイプ結合するときには、署名付きのキーを使用してライセンスを付与する必要があります。

Token ライセンス

トークンを使用してアクセス パスをユーザーに割り当てて、ユーザーが Qlik Sense にアクセスできるようにします。License Enabler File (LEF) によって割り当て可能なトークンの数が指定され、サイトのセントラル ノードで使用可能なトークンの数が保持されます。つまり、Qlik Sense サイトは少なくとも 1 つの LEF を必要としています。アクセス パスなしのユーザーはアプリにアクセスできません。

この Token ライセンスは、既存の Qlik Sense Token ライセンスを持つ顧客にのみ付与できます。

Qlik Sense で QlikView CAL ベースのライセンスを使用することはできません。これは、トークンに QlikView で使用される Client Access ライセンス (CAL) との互換性がないためです。

トークン数の増加

(トークンの追加購入などで)LEF のトークン数が増加した場合、新しいトークンは、Qlik Sense にアクセス権のあるユーザーにアクセス パスを割り当てるために使用できる、未割り当てトークンのプールに追加されます。

トークン数の減少

LEF のトークン数が減少した場合、以下が行われます。

1. 未割り当てトークンを削除します。
2. 手順 1 では LEF の減少したトークン数を補充できない場合、アクセス パスを削除して解放したトークンは、割り当てたトークンの数が LEF に新しく設定した数を下回るまで、新規割り当てとして使用できません。

アクセスの割り当て

Qlik Sense Enterprise ライセンスは、アクセス タイプまたは トークンのいずれかに基づいています。ライセンスに応じて、アクセス タイプまたはアクセス パスをユーザーに割り当てて、Qlik Sense にアクセスできるようにします。

- アクセス タイプのライセンスは、Professional User とAnalyzer User ライセンス (ユーザーベース) および Analyzer Capacity ライセンス (キャパシティベース) です。
  Professional User および Analyzer User ライセンスは、Professional アクセスおよび Analyzer アクセスを割り当てることができます。
  Analyzer Capacity ライセンスでは、Analyzer Capacity ライセンスを割り当てることができます。ここでは消費が時間ベース (Analyzer タイム) となっています。
- Qlik Sense Token ライセンスでは、トークンを使用してアクセス パスをユーザーに割り当てます。User アクセスおよび Login アクセスを割り当てることができます。

アクセス タイプ

Professional User および Analyzer User ライセンスとAnalyzer Capacity ライセンスは、アクセスの既定数の割り当てを許可します。License Enabler File (LEF) で、ライセンスの契約条件およびユーザーに割り当て可能なアクセス タイプを定義します。展開とアクティブ化を行う際に署名付きライセンス キーを使用した場合、サブスクリプション ベースのライセンスとしてこうしたライセンスを組み合わせることも可能です。永続ライセンスを使用している場合は、ユーザーベースのライセンスのみを組み合わせることができます。Analyzer Capacity アクセス権のライセンスを付与する場合は、署名付きキーによるライセンスを使用する必要があります。

Professional アクセス

識別されたユーザーに Professional アクセス権を割り当て、Qlik Sense サイト内のストリームとアプリにアクセスできるようにします。プロフェッショナル アクセスは、Qlik Sense インストールのすべての機能にアクセスする必要があるユーザーが対象です。プロフェッショナル アクセス権のあるユーザーは、シートまたはアプリを作成、編集、公開でき、Qlik Sense サイトの管理などの使用可能な機能をすべて使うことができます。

シルアル番号とコントロール ナンバーでライセンスが付与された Qlik Sense インストールでは、ユーザーから Professional アクセスの割り当てを削除すると、アクセス権が直近 7 日間で使用されていた場合、そのアクセス タイプは検疫のため隔離されます。直近 7 日間で使用されていない場合、プロフェッショナル アクセス権は直ちに解放されます。7 日以内であれば、同一のユーザーに検疫状態のプロフェッショナル アクセス権を復元することができます。

署名付きのキーを使用してライセンスが付与されている Qlik Sense インストールには、検疫は適用されません。

このアクセス パスのタイプでは、単一ユーザーが行える並列ユーザー接続の最大数は、5 回です。署名付きライセンス キーでライセンスを使用する場合、QMC にアクセスすると、並列セッションの最大数がカウントされて追加され、5 つとなります。セッションの不必要な消費を避けるには、ルート管理者にどのタイプのアクセスも割り当てないでください。

最大数の並列ユーザー接続を持つユーザーが接続を終了する (たとえば、ログアウト) 場合、別の接続を追加する (たとえば、ログイン) には、アクセス パスが使用可能になる前に 5 分が経過する必要があります。

アナライザー アクセス

識別されたユーザーに割り当てられた Analyzer アクセス権によって、ハブ内のストリームとアプリにアクセスできるようにします。アナライザー アクセスは、他のユーザーによって作成されたシートとアプリを使用するユーザーが対象です。アナライザー アクセス権を持つユーザーは、シートやアプリを作成、編集、または公開することはできませんが、アプリ内のデータに基づいて、ストーリー、ブックマーク、スナップショットを作成できます。また、ユーザーは、ブックマーク、印刷オブジェクト、ストーリー、シートを作成したり、オブジェクトから Excel にデータをエクスポートしたりすることもできます。

シリアル番号とコントロール ナンバーでライセンスが付与された Qlik Sense インストールでは、ユーザーから Analyzer アクセスの割り当てを削除すると、アクセス権が直近 7 日間で使用されていた場合、そのアクセス タイプは検疫のため隔離されます。直近 7 日間で使用されていない場合、アナライザー アクセス権は直ちに解放されます。7 日以内であれば、同一のユーザーに検疫状態のアナライザー アクセス権を復元することができます。

署名付きのキーを使用してライセンスが付与されている Qlik Sense インストールには、検疫は適用されません。

このアクセス パスのタイプでは、単一ユーザーが行える並列ユーザー接続の最大数は、5 回です。最大数の並列ユーザー接続を持つユーザーが接続を終了する (たとえば、ログアウト) 場合、別の接続を追加する (たとえば、ログイン) には、アクセス パスが使用可能になる前に 5 分が経過する必要があります。

Analyzer Capacity アクセス

アナライザー キャパシティは消費ベースのライセンス タイプで、使用可能な機能に関してアナライザー アクセスに似ています。ユーザーは、ハブ内のストリームとアプリにアクセスし、他のユーザーによって作成されたシートとアプリを消費できます。Analyzer Capacity アクセスでは、アプリ内のデータに基づいてストーリー、ブックマーク、スナップショットを作成し、公開することができます。シートまたはアプリの作成、編集、または公開はできません。

アナライザー キャパシティ ライセンスにより、アナライザー時間 (毎月の定義済み分時間 (カレンダー日付)) に対してサブスクライブします。これらの分時間は、ユーザー間で共有され、匿名ユーザーを含むユーザー グループの一員になっている任意のユーザーが消費可能です。消費は 6 分単位で測定されます。新しい 6 分間ごとに、1 単位が消費されます。

アクセス パス

Qlik Sense Token ライセンスでは、トークンを使用してアクセス パスをユーザーに割り当てます。License Enabler File (LEF) により、さまざまなアクセス パスに割り当てることができるトークンの数が決まります。アクセス パスなしのユーザーはアプリにアクセスできません。

ユーザー アクセス パス

このタイプのアクセス パスでは、一意に指定されたユーザーがハブにアクセスできます。

アクセス パスは、Qlik Sense サイト全体で有効です。たとえば、ユーザーがまず米国のノードに接続して、その後英国のノードに接続した場合、2 つのノードが同じセントラル ノードに接続されているなら、ユーザーは同じアクセス パスを使用します。

このアクセス パスのタイプでは、単一ユーザーが行える並列ユーザー接続の最大数は、5 回です。最大数の並列ユーザー接続を持つユーザーが接続を終了する (たとえば、ログアウト) 場合、別の接続を追加する (たとえば、ログイン) には、アクセス パスが使用可能になる前に 5 分が経過する必要があります。

1 個のトークンは、1 個のユーザー アクセス パスに対応します。アクセス パスは、Qlik Management Console (QMC) を使用して割り当てられます。

> *ユーザー アクセス パスと、ログイン アクセス パスを消費する可能性の両方があります。アクティブ セッションが 5 つある場合は、追加のセッションを開くとログイン アクセス パスから消費されます。*

User アクセス パスの割り当ての削除

ユーザー アクセス パスが削除されると、最後にアクセス パスが使用された時から数えて 7 日間の検疫期間に入ります。たとえば、アクセス パスが 1 月 10 日に使用された場合、そのアクセス パスを割り当てるために使用されたトークンは、1 月 18 日まで新規の割り当てのためには使用できません。検疫期間中、アクセス パスの元の割り当てが復帰します。つまり、検疫期間が終了すると、ユーザーは再びアクセス パスの使用を開始できます。

ログイン アクセス パス

このタイプのアクセス パスでは、指定されたユーザーあるいは匿名ユーザーが、28 日間につき最大 60 分間続けてハブにアクセスできます。ユーザーが 60 分の制限を超えても、ユーザー接続はタイムアウトしません。その代わり、別のログイン アクセス パスが使用されます。利用できるログイン アクセス パスがこれ以上ない場合、ユーザー接続は終了します。

- 認識されたユーザーが切断された場合、60 分以内に再接続すれば、そのユーザーは再接続して同じアクセス パスを使用できます。
- 匿名のユーザーが切断された場合、そのユーザーは再接続時に新しいアクセス パスを取得します。

ログイン アクセス パスはログイン数を記録し、28 日間にわたり作動します。たとえば、グループ A に 1000 回のログイン数が割り当てられている場合、グループ A のユーザーは、28 日間にわたって 1000 回のログイン数を利用できます。ログイン数 100 回分が第 1 日目に使用された場合でも、第 29 日目に再び、100 回のログイン数を利用できます。

このアクセス パスのタイプでは、単一ユーザーが行える並列ユーザー接続の最大数は、5 回です。これは、認識されたユーザーにのみ適用されることにご注意ください。匿名のユーザーが持つことのできるユーザー接続は 1 つのみです。最大数の並列ユーザー接続を持つユーザーが接続を終了する (たとえば、ログアウト) 場合、別の接続を追加する (たとえば、ログイン) には、アクセス パスが使用可能になる前に 5 分が経過する必要があります。ただし、追加のアクセス パスを消費すれば、1 つのアクセス パスで許可されている数を超える接続を使用できます。

1 個のトークンは、10 個のログイン アクセス パスに対応します。アクセス パスは、QMCのログイン アクセス グループを使って割り当てられます。

アプリがアクティブに使用されていない場合、アプリをリロードするとセッションが延長され、アクセスパスも消費されます。アプリとともにブラウザーページが開いている場合、アプリをリロードするとさらにアクセスパスが消費されます。

Login アクセスパスの割り当ての削除

ログインアクセスグループが削除されると、アクセスパスを割り当てるために使用されていたトークンは、以下の手順に従い使用可能となります。

1. 未**使用の**ログインアクセスパス10個に付き、1個のトークンを解放します。
2. 前述の *アクセスの割り当て (page 12)* セクションで指定されている期間終了後も**使用中**のステートのままであるLoginアクセスパスは、10個につき1個のトークンを解放します。

1.2 インストールする前に Qlik Sense Enterprise on Kubernetes

Qlik Sense Enterprise on Kubernetes展開を正しく計画して準備するには、次のことを行います。

システム要件
: まず、システム要件を満たしていることを確認します。

対応ブラウザ
: ブラウザーがサポートされていることを確認します。

マルチクラウドサービス
: Qlik Senseサービスを理解します。

Qlik Sense Enterprise on Kubernetes のシステム要件

このセクションでは、Qlik Sense Enterprise on Kubernetesを正常にインストールし稼働させるために、対象システムが満たす必要がある要件を記載しています。

Kubernetes 環境	Kubernetes環境には、Qlik Helmおよびコンテナーイメージリポジトリへのインターネットアクセスが必要です。 Kubernetesサービスベンダー: • Microsoft Azure使用 Azure Kubernetes Service (AKS) • Amazon Web Services (AWS)使用 Amazon Elastic Container Service for Kubernetes (EKS) • Amazon Web Services (AWS)を介して展開 Kubernetes Operations (KOPs) • Google Cloud使用 Google Kubernetes Engine (GKE) • Red Hat OpenShift 4+

	管理対象外 Kubernetes 展開: • Kubernetes v1.10.xを超えv1.16.x未満のクラスタ
Kubernetes パッケージ マネージャー	Helm v2.12.0を超えv2.15.x未満
ローカル/評価/テスト環境	Windows:Minikube v0.33 + Red Hat MiniShift v1.21.0+ Mac:Kubernetes が有効化された Desktop の Docker: v2.0.0.3
データベース	MongoDB 3.6+
ファイル システム	ReadWriteMany に対応したクラスターに接続されているストレージ。これは、Storage Class または Persistent Volume Claim として設定可能です。
プロセッサ (CPU)	最小 4 コア (データ量に応じて追加)
メモリ	最小 8 GB (データ量に応じて追加)
ディスク空き容量	インストールには計 5 GB が必要
IDP	ユーザー認証の場合は OIDC 互換 IDP が必要

Qlik Sense Enterprise on Kubernetes 対応ブラウザ

Qlik Sense は、デフォルトのブラウザ設定を使用して、このセクションに記載されているプラットフォームとWeb ブラウザの組み合わせで動作するよう設計されています。

Qlik Sense の各リリースは、公開されている最新のブラウザー バージョンとの互換性についてテストされています。ブラウザーのバージョンは頻繁に更新されるため、Qlik のシステム要件にはブラウザーの特定のバージョン番号は含まれていません。

各 Qlik Sense リリースは、Qlik Sense リリースの時点で公開されている最新の iOS バージョンと互換性があり、その最新バージョンでサポートされています。iOS のバージョンは頻繁に更新されるため、Qlik のシステム要件には iOS の特定のバージョン番号は含まれていません。

Microsoft Windows 7

- Microsoft Internet Explorer 11
- Google Chrome
- Mozilla Firefox (ハードウェア アクセラレーションが必要。仮想環境には対応していません)

Microsoft Windows 8.1

- Microsoft Internet Explorer 11
- Google Chrome
- Mozilla Firefox (ハードウェア アクセラレーションが必要。仮想環境には対応していません)

Microsoft Windows 10

- Microsoft Edge
- Microsoft Internet Explorer 11

- Google Chrome
- Mozilla Firefox (ハードウェア アクセラレーション が必要。仮想環境には対応していません)

Apple Mac OS X 10.11 および 10.12

- Apple Safari 10 以降
- Google Chrome
- Mozilla Firefox (ハードウェア アクセラレーション が必要。仮想環境には対応していません)。

Microsoft Windows Server 2012 R2

- Microsoft Internet Explorer 11
- Google Chrome
- Mozilla Firefox (ハードウェア アクセラレーション が必要。仮想環境には対応していません)

Microsoft Windows Server 2016

- Microsoft Internet Explorer 11
- Google Chrome
- Mozilla Firefox (ハードウェア アクセラレーション が必要。仮想環境には対応していません)

CefSharp 埋め込みブラウザー v55 以降 (CefSharp を使用すると、Chromium オープン ソース ブラウザーを .Net アプリ内に組み込むことができます)

Microsoft Windows Server 2019

- Microsoft Internet Explorer 11
- Google Chrome
- Mozilla Firefox (ハードウェア アクセラレーション が必要。仮想環境には対応していません)

CefSharp 埋め込みブラウザー v55 以降 (CefSharp を使用すると、Chromium オープン ソース ブラウザーを .Net アプリ内に組み込むことができます)

iOS

バージョン 11.2 以降 (タブレット デバイスでは、スクリプト編集には対応していません)

Qlik Sense バージョン:Qlik Sense EnterpriseSeptember 2017 以降

対応デバイス:

- iPad Air 以降
- iPhone 5S 以降

対応ブラウザ:

- Apple Safari
- Google Chrome
- VMware ブラウザ (AirWatch アプリ毎 VPN を使用)

- BlackBerry Access 2.9.1 以降 (BlackBerry Dynamics プラットフォームを使用)

BlackBerry Access ブラウザーを使用するには iOS 11.3 が必要です。

Android

バージョン 6.0、7.1、8.1、9.0 (タブレットデバイスではスクリプト編集に対応していません):

- Google Chrome
- BlackBerry Access 2.9.1 以降 (BlackBerry Dynamics プラットフォームを使用)

Windows 10 phone

- Microsoft Edge

最小画面解像度は、デスクトップとラップトップでは 1024x768、タブレットでは 1024x768、小型画面では 320x568 です。

マルチクラウド サービス

Qlik Sense Enterprise on Windows 環境を展開する場合、複数のオプションがあります。マルチクラウド展開内で実行する必要があるサービスは、次のように分類できます。

通常、QCS 展開内で実行されるサービスは、Qlik Sense Enterprise on Kubernetes 展開内で実行されるサービスと似ていますが、アクセスはできません。これは、Qlik がインフラストラクチャを管理しているからです。QCS SaaS に接続できても、Kubernetes 展開と同じ構成オプションはありません。

Windows 展開上のサービス

Qlik Sense Enterprise on Windows 展開内でマルチクラウド機能を使用する場合は、以下に示すサービスが必要です。

サービス	説明
App Distribution Service	ポリシーベースのアプリ配布ルールに基づいて、アプリおよび関連付けられたメタデータを定義済みの配布ターゲットに配布します。
Hybrid Deployment Service	すべてのターゲット環境の資格情報とURL を含む構成の詳細をマルチクラウド展開に保存します。
Hybrid Setup Console Service	マルチクラウド展開内に構成されたターゲット環境を管理するための (資格情報とサービス URL を含めて) Multi-cloud Setup Console UI 機能。
Resource Distribution Service	インストール済みの拡張機能とテーマを各クラウド環境のリソース ライブラリに公開します。

Kubernetes および Qlik Cloud Services 展開でのサービス

Qlik Sense Enterprise on Kubernetes で実行するサービスは、展開の要件に応じてさまざまに異なります。

サービス	説明
Chronos	スケジューラ バック エンド サービス。
クラウド ハブ	Qlik Cloud Services とQlik Sense Enterprise on Kubernetes のユーザーに対するハブの機能を処理します。
コレクション	ハブに送られたコンテンツを編成し、構造化します。また、アクセス制御ルールにも適用されます。
datafiles	アプリのロード中にアクセス可能なデータ ファイルのアップロード/管理をユーザーに許可します。
edge-auth	外部 ID プロバイダーと連携して、展開へのログイン時にユーザーを認証するサービス。内部リソースへの安全なアクセスを認証するチケットも管理します。
elastic-infra	MongoDB、Redis、Traefik、ngingx-ingress といった非 Qlik サービスを含んだコレクション。MongoDB、Redis、nginx-ingress。Helm パッケージ マネージャーを使って、Kubernetes クラスター上で elastic-infra 展開を起動します。クラウド環境内で必要なすべてのコンポーネントと機能を接続するための基本リソースを起動します。
エンジン	すべてのアプリケーションの計算とロジックを処理します。
機能フラグ	高度なシナリオにおいて機能のオンとオフを切り替えます。
洞察	シートの [共有] ダイアログを支援する機能に関する REST サービス。このサービスは、共有リソースへの持続的な固定リンクの生成、追跡、および供給を行うことで、Qlik Sense 洞察の共有を担います。固定リンクは、さまざまなソーシャル メディアで共有できます。
ライセンス	ラインセンス サービスは、Qlik Cloud Services とQlik Sense Enterprise on Kubernetes でユーザー ライセンスを供与するために使用されます。
ロケール	クライアントでのユーザー ロケールの選択を処理します。
Mira	展開内のエンジン、その現在の正常性、アプリケーションの可用性について検出サービスを提供します。
ODAG	Qlik Sense アプリの On-demand アプリ生成用のサービス。
Policy Decision	Qlik オブジェクト(アプリなど) に対して ABAC セキュリティ評価を実行する Qlik Cloud Services および Qlik Sense Enterprise on Kubernetes で一連のルールを処理します。Rules Service と呼ばれることもあります。ルール エンジン用に REST API を、ルールベースのポリシー用に管理 API を使用し、QRS ルール エンジン、Policy Decision Service を置き換えます。
qix-data-connection	接続管理に関するエンジン リクエストを処理します。
qix-sessions	ユーザー セッション トラフィックをエンジン サービスにルーティングします。
レポート作成	データおよびチャート画像を含むレポートの生成を実行します。
リソース ライブラリ	テーマや拡張機能などのコンテンツをサポートするための、汎用リソース ストレージ サービス。
テナント	テナント(ユーザー) 情報を保存し、返すために使用されます。

temporary-contents	一時的に使用可能になっているリソースを管理します。
ユーザー	ユーザー情報を管理及び取得します。
Sense Client	Qlik Sense Enterprise では開発者により、QCS および Qlik Sense Enterprise on Kubernetes では消費者によって実行される、Qlik Sense クライアントのデスクトップおよび Web ブラウザー インスタンス。

1.3 Qlik Sense Enterprise on Kubernetes の展開

この図は、1 つの Kubernetes クラスターが Qlik Sense Enterprise on Windows ノードに接続されている Qlik Sense Enterprise on Kubernetes 展開の例を示しています。このクラスターには、一連のノードに展開されたエンジン サービスや他のサービスなど、1 つ以上の Qlik Sense マイクロサービスが含まれています。この展開では、多数のアプリ(読み取り専用) をスケールアップしてユーザーに提供できます。パブリックまたはプライベート クラウド内に展開された Kubernetes クラスターは、データ容量と MongoDB インスタンスを共有します。ID プロバイダー (IdP) はユーザーを認証し、QSE は組み込みセキュリティルールを使ってマルチクラウド アプリへのアクセスを承認します。IdP により、セキュリティルール対象の Qlik Sense Enterprise とクラウド環境のコンテンツへの、同じ名前のユーザーによるアクセスが許可されます。Kubernetes クラスター、パブリックまたはプライベート クラウド、ネットワーク インフラストラクチャはすべてユーザーが管理します。

Qlik Sense Enterprise on Kubernetes 向けの CSRF セキュリティ

クロスサイト リクエスト フォージェリ(CSRF) とは、第三者がユーザー認証情報を悪用して、ユーザーの Web アプリケーションを攻撃することです。例えば、セキュアな Web アプリケーションで既に認証されているユーザーが、Web セッションの実行中に悪意のあるリンクをクリックすると、攻撃者は、そのユーザーの許可や情報がなくともユーザーの認証情報を使ってタスクやアクションを実行することができます。

Qlik Sense Enterprise on Kubernetes の API を CSRF のセキュリティリスクから確実に保護するため、CSRF 攻撃を防ぐトークン ベースの CSRF 対策セキュリティを Qlik に導入しました。

このトークンはサーバー側で生成され、Web サーバーによって特定のセッションとリンクされます。その後、隠れた値として、すべてのWeb アプリケーション フォームで使用されます。トークンは Web セッションではなくサーバー側に存在するため、サーバーへのアクセス権限がないハッカーはそのトークンを取得することができません。

2 Qlik Sense Enterprise on Kubernetes の準備

Qlik Sense Enterprise on Kubernetes は、Helm チャートとして提供されるパッケージ内のコンテナ画像のセットの形態で Kubernetes クラスターに展開されます。インストールできるようにするには、最低でも次のアイテムが所定の位置に配置されている必要があります (詳しくは、システム要件 も参照してください)。

- 稼働中の Kubernetes クラスター - 開発目的でローカルに実行するか、AWS、Google Cloud、Azure などのクラウドベンダーに展開することができます。
- Kubernetes クラスターには、データを持続的に保存するためのファイル ストレージへのアクセス権が必要です。これは、**`readwritemany`** アクセスを許可するストレージ クラスとして提供される必要があります。インストール時には、ストレージ クラスの名前が必要になります。
- また、Qlik Sense Enterprise のライセンス キーも必要とされ、このキーは署名付きライセンス キー形式になっている必要があります。シリアル番号とコントロール ナンバーは使用できません。
  このバージョンのライセンスを持っていない場合は、Qlik サポートにお問い合わせください。

Qlik のドキュメントには Kubernetes クラスターのインストールと構成について述べられていないため、これについては、https://Kubernetes.io で、または使用しているクラウドベンダーまたはクラウド製品のドキュメントを確認する必要があります。

Kubernetes 環境とやり取りし、コマンドを実行し、ソフトウェアを展開するには、次のツールをローカル マシンにインストールしておくことも必要です。

- Kubectl - 管理コマンドを実行するマシンに kubectl をインストールします。使用しているオペレーティング システムの詳細については、https://Kubernetes.io を参照してください。

  > *複数のクラスターがある場合、このソフトウェアはさまざまなクラスターを対象にコマンドを実行できます。コマンドが正しい Kubernetes インスタンスに対して発行されるようにします。*

- Helm - Helm は Kubernetes のパッケージ マネージャーです。チャートと呼ばれる概念があり、必要とされるサービス、使用されるイメージ、Kubernetes クラスターで実行するときの既定の設定を定義するために使用されます。Qlik Sense パッケージを Kubernetes にプッシュするために使用され、kubectl に依存しているため、kubectl と同じマシンにインストールする必要があります。Qlik は、helm を使用して既定のチャートを定義して、ユーザーが簡単に展開できるようにしています。
  ローカル マシンに Helm をインストールするには、https://docs.helm.sh/ にある使用しているオペレーティング システムの指示に従ってください。

2.1 ローカル ツールの準備

Kubernetes クラスターをセットアップしたら、Kubernetes クラスターで使用するローカル ツールを準備する必要があります。ローカル ツールを準備するには、次の操作を行う必要があります。

- kubectl を Kubernetes クラスターにバインドする
- Qlik の helm チャート リポジトリを追加する
- Kubernetes クラスターで使用する helm を初期化する

作業を開始する前に、ローカル マシンに以下をインストールしておく必要があります。

- Kubectl
- Helm

kubectl を Kubernetes クラスターにバインドするには:

1. 次のコマンドを使用して、kubectl が Kubernetes クラスターを指していることを確認します。
   ```
   kubectl config current-context
   ```
2. kubectl が Kubernetes クラスターを指していない場合は、次のコマンドを使用して使用可能なクラスターのリストを取得します。
   ```
   kubectl config get-clusters
   ```
3. 次のコマンドを使用して、適切なクラスターを指すように kubectl を設定します。
   ```
   kubectl config set-cluster <cluster-name>
   ```

Qlik の helm チャートリポジトリを追加するには:

1. 次のコマンドを実行して Qlik の helm チャートリポジトリを Helm に追加します。これは、Qlik Sense がプルされる場所です
   ```
   helm repo add qlik https://qlik.bintray.com/stable
   ```
2. 次のコマンドを使用して、構成されているすべてのリポジトリのリストを取得し、Qlik helm チャートリポジトリが正常に追加されたことを確認します。
   ```
   helm repo list
   ```

helm を Kubernetes クラスターで使用するには、初期化して、インストールを処理する helm Tiller pod を作成する必要があります。

1. 単純な場合には、次のコマンドを使用して作成します
   helm を使用して Kubernetes に展開するため、helm Tiller pod は最初に Kubernetes クラスターに追加されます。
   単純な場合には、次のコマンドを使用して追加します:
   ```
   helm init --wait
   ```
2. Kubernetes クラスターで RBAC などのセキュリティ機能が有効になっている場合は、追加で次のコマンドを実行する必要があります。
   ```
   kubectl create serviceaccount --namespace kube-system tiller
   kubectl create clusterrolebinding tiller-cluster-rule --clusterrole=cluster-admin --
   serviceaccount=kube-system:tiller

   helm init --upgrade --wait
   kubectl patch deploy --namespace kube-system tiller-deploy -p '{"spec":{"template":{"spec":
   {"serviceAccount":"tiller"}}}}'
   ```

3 Qlik Sense Enterprise on Kubernetes のインストール

Kubernetes クラスターをセットアップし、ローカル ツールを用意すれば、Qlik Sense Enterprise を Kubernetes クラスターにインストールする準備が整います。

要約すると、インストール前に最低でも以下の作業を終了しておきます。

- Kubernetes クラスターをセットアップし、**`readwritemany`** ストレージのストレージ クラスを追加する
- ローカル ツールを Kubernetes クラスターで使用できるように準備する

以下に、最初の簡単なインストールのインストール ステップを示します。これには、稼働に必要な MongoDB のテスト インスタンスと単純な IDP の展開が含まれます。実稼働可能な状態に移行するため、以下の領域についての追加のトピックをもう一度参照してください。

- ユーザーを認証するための IDP の理解と構成
- 別個の MongoDB インスタンスの設定
- ログの表示と処理

3.1 設定値の入力

Qlik Sense Enterprise on Kubernetes のインストール時には、2 つの方法でインストーラに設定値を指定できます。

- `helm install` コマンドのパラメーターとして指定する。
- **values.yaml** の設定を参照し、その情報を `helm install` コマンドで使用する。

構成設定を **values.yaml** に保存することにより、複数の展開でその設定を再使用し、簡単に新しい config セクションを追加できます。これにより、バージョン管理も可能です。

values.yaml ファイルの情報は、主に Qlik ヘルプで使用されます。YAML ファイルに関する詳細な情報は、https://en.wikipedia.org/wiki/YAML などのサイトでオンラインで参照できます。

3.2 Qlik Sense のインストール

以下のステップに従って Qlik Sense をインストールします。

1. **values.yaml** というファイル名でテキスト ファイルを作成します。
   a. 以下の内容をファイルに追加します。

```
#This setting enables dev mode to include a local MongoDB install
devMode:
  enabled: true

#This setting accepts the EULA for the product
engine:
  acceptEULA: "yes"
```

devMode.enabled を true に設定すると、MongoDB インスタンスは、展開とテストの目的でのみ Kubernetes での Qlik Sense Enterprise on Kubernetes の内部に展開されます。

b. ストレージを必要とするサービスを指し示す次の内容を Kubernetes ストレージ クラスに追加し、必要に応じてストレージ クラスの名前をを更新します。

Kubernetes via Docker for Desktop または Minikube を使用する場合は、このセクションを追加しないでください。

```
#These setting specifies the storage for the services
global:
  persistence:
    storageClass: my-storage-class
```

c. テスト目的で Minikube を使用している場合は、ここで追加の構成を確認して追加する必要があります。*Minikube の使用 (page 25)*を参照してください。

d. ファイルを保存します。

2. Qlik Sense はエンジンを動的に作成してスケジュールされたロードを実行します。この動作を可能にするには、エンジンをカスタム リソースとして Kubernetes で構成する必要があります。
このステップは、1 つのクラスターについて 1 回だけ行う必要があります。次のコマンドを実行して、動的エンジンが使用するカスタム リソース定義をインストールします。
```
helm install --name qliksense-init qlik/qliksense-init
```

3. 次のステップでは、Qlik Sense パッケージをインストールします。次のコマンドを実行します。
```
helm install -n qliksense qlik/qliksense -f values.yaml
```
これで、すべての画像のダウンロードとそれらの実行を含め、Kubernetes クラスターでの展開がソフトウェアで開始されます。

4. この段階で、kubectl を使用して進行状況を確認できます。次のコマンドを実行します。
```
kubectl get pods
```
正常に展開されると、次のようなテキストが表示されます。
```
NAME                                                READY     STATUS    RESTARTS   AGE
qliksense-collections-7f456595b8-vjhtf              1/1       Running   0          2m
qliksense-edge-auth-858f89b849-42z66                2/2       Running   0          2m
… (lines removed for brevity)
```

通常、初期化されて「実行中」のステータスが表示されるまでに数分の時間がかかります。

*サービスが開始されない場合は、そのサービスのログ ファイルで詳細を確認してください。ペンディング状態のサービスが残っている場合は、Kubernetes クラスターにストレージ クラスとして **`readwritemany`** ストレージが存在しており、YAML で正しく参照されていることを確認してください。*

3.3 展開へのアクセス

ハブに接続し、Kubernetes クラスター内部のインストールの URL を取得するために必要なインストールを確認します。これは、構成またはベンダーによって次のように異なります。

ほとんどのクラウドベンダー (AWS、GCP、Azure など) では、インストール時に IP アドレスが自動的に生成されます。アドレスは、次のコマンドを実行して調べることができます。

```
kubectl describe service qliksense-nginx-ingress-controller
```

Docker Desktop の場合、IP は通常、マシンのループバックアドレス `127.0.0.1` です。Minikube の場合は、`minikube ip` コマンドを実行することによって IP を入手できます。

IP アドレスへのエイリアスの作成

この簡単な展開では、サンプルの ID プロバイダーが自動的に設定されます。これにより、いくつかのサンプル アカウントで hub にログインすることができます。このサービスは、既定で URL *https://elastic.example* でのみ待ち受け、IP アドレスだけでは参照することができません。このため、ホスト ファイル エントリを追加し、その URL からの IP アドレスを *elastic.example* のエイリアスに指すように指定する必要があります。

このステップは、基本例を実行する場合にのみ必要とされ、実稼働では実際の ID プロバイダーが使用され、その期間にはクラスターの正しい DNS エントリが使用されます。

> *シンプル IdP はテスト目的でのみ使用されており、ID プロバイダーの設定 (page 29) を再び参照してフル IdP を設定する必要があります。*

ログインとライセンスの適用

これで、*https://elastic.example* において hub を参照することができるようになりました。ログインを求められますが、サンプル アカウントとして **harley@qlik.example** を、パスワードとして **Password1!** を使用できます。

アプリケーションを作成できるようにするには、ここで *https://elastic.example/console* に移動してライセンスを適用する必要があります。

3.4 Minikube の使用

Minikube は、Kubernetes 環境が Windows、Mac、または Linux マシンでローカルに実行されることを許可するテスト環境です。ローカル テストの目的で Qlik Sense Enterprise on Kubernetes を実行するために使用できます。

Minikube は他のすべての Kubernetes プロバイダーが使用する LoadBalancer リソースに対応しておらず、別のポートで実行する必要があり、使用する場合には追加の構成セクションが必要になります。

> *これらの追加の構成アイテムは、使用できないので他の Kubernetes プロバイダーでは使用しないでください。*

Minikube の準備

準備を開始する前に、https://kubernetes.io/docs/setup/minikube/ で、使用するオペレーティングシステムに合った Minikube のインストール ドキュメントを再度お読みください。

Minikube の起動時には、以下のスクリプトを実行して十分なリソースがあることを確認してください (`minikube delete` を最初に実行する必要があります)。

```
minikube start --memory 8000 --cpus=2
```

Minikube を使用する場合の追加構成

Minikube の実行中に、*Qlik Sense Enterprise on Kubernetes のインストール (page 23)* のインストール手順に従って操作します。ただし、**values.yaml** に次の追加セクションを追加してください。

```
# MINIKUBE SPECIFIC SETTINGS (dont not use with other K8 providers)_____________

elastic-infra:
 nginx-ingress:
   controller:
     service:
       type: NodePort
       nodePorts:
         https: 32443
     extraArgs.report-node-internal-ip-address: ""

hub:
 ingress:
   annotations:
     nginx.ingress.kubernetes.io/auth-signin: https://$host:32443/login?returnto=$request_uri

management-console:
 ingress:
   annotations:
     nginx.ingress.kubernetes.io/auth-signin: https://$host:32443/login?returnto=$request_uri

edge-auth:
  oidc:
    redirectUri: https://elastic.example:32443/login/callback
```

Minikube のインストールへのアクセス

起動したら、次のコマンドを使用して、Kubernetes クラスターが実行されている IP アドレスを取得できます。

```
minikube ip
```

Minikube の使用時には、製品 IP へのアクセスに使用される URL で 32443 ポートを指定する必要があります (例、https://elastic.example:32443)。

3.5 Red Hat OpenShift プラットフォームへの Qlik Sense Enterprise on Kubernetes のインストール

Qlik Sense Enterprise を Red Hat OpenShift Kubernetes プラットフォームにインストールするために、追加して考慮すべき事項は以下のとおりです。

- OpenShift の既定の namespace は **myproject** です。
- OpenShift は通常、独自の Docker 画像レジストリを備えています。現在すべてのチャートが global.imageRegistry に対応しているので、顧客は自分の画像レジストリからすべての画像プルをポイントすることができます。
- OpenShift では、root として実行するコンテナーは既定では許可されていません。
- OpenShift はクラスター全体のアクセスを阻止します。
- OpenShift は、サービスがルートによって公開されることを期待します。
- OpenShift は独自の認定コンテナー レジストリを備えており、認定されていないコンテナーを阻止します。
- OpenShift 展開では、通常はインータネット アクセスは必要ありません。Qlik Sense Enterprise on Kubernetes ではインータネット アクセスが必要です。

Kubernetes クラスターをセットアップし、ローカル ツールを用意すれば、Qlik Sense Enterprise を Kubernetes クラスターにインストールする準備が整います。「*Qlik Sense Enterprise on Kubernetes のインストール (page 23)*」を参照してください。

> *Qlik は Red Hat OpenShift プラットフォームへの Qlik Sense Enterprise on Kubernetes の展開に対応していますが、この展開は現在 Red Hat では認定されていません。*

MiniShift の使用

OpenShift プラットフォームを実行させるには、MiniShift を使用して仮想マシン (VM) を作成し、その VM にローカルのシングル ノード OpenShift クラスタをプロビジョニングします。RedHat では、OpenShift プラットフォームを実行するために、MiniShift をシンプルとして提供しています。MiniShift は https://github.com/minishift/minishift で入手できます。

1. MiniShift の最新バージョンをダウンロードします。
2. これをディレクトリに解凍して、このディレクトリをパスに追加します。
3. helm (必須) および kubectl (オプション) をインストールします。
4. MiniShift のリソース設定を行い、Qlik Sense Enterprise on Kubernetes を完全に展開できる十分なリソースがあることを確認します。次のコマンドを使用します。
   ```
   minishift config set memory 8192
   minishift config set cpus 4
   ```
5. 次のコマンドを使用して MiniShift を起動します。
   ```
   minishift start --vm-driver=virtualbox
   ```
6. 次のコマンドを使用して環境を構成します。
   ```
   minishift oc-env
   ```
7. 新しいプロジェクトを作成します。プロジェクトは、Kubernetes namespace の OpenShift と同義のもので

す。

この例では、kubectl の代わりに oc を使用することに注意してください。ほとんどの場合これらは同じ内容を実行しますが、oc は、kubectl ではできないことをいくつか実行します。

```
oc new-project tiller
```

8. tiller が見つかる場所を helm が分かるように変数を設定します。
```
$Env:TILLER_NAMESPACE="tiller"
```
9. helm クライアントを設定します。
```
helm init --client-only
```
10. tiller を OpenShift にインストールするには特殊な処理が必要です。OpenShift には、それを実行するための yaml ファイルが備えられています。次のコマンドを実行します。
```
oc process -f https://github.com/openshift/origin/raw/master/examples/helm/tiller-template.yaml -p TILLER_NAMESPACE="$Env:TILLER_NAMESPACE" -p HELM_VERSION=v2.9.1 | oc create -f -
```
11. 準備ができていることを確認するため、次のコマンドを実行します。
```
oc rollout status deployment tiller
```
12. 次のコマンドを使用して、Qlik 展開のプロジェクトを作成します。
```
oc new-project qlik
```
13. セキュリティを設定するため、次のコマンドを使用してシステム管理者としてログインする必要があります。
```
oc login -u system:admin
```
14. 次のコマンドを実行して、必要なセキュリティ設定を行います。
```
oc policy add-role-to-user edit "system:serviceaccount:$($Env:TILLER_NAMESPACE):tiller"
oc policy add-role-to-user cluster-admin "system:serviceaccount:$($Env:TILLER_NAMESPACE):tiller"
oc adm policy add-scc-to-user anyuid "system:serviceaccount:$($Env:TILLER_NAMESPACE):tiller"
oc adm policy add-cluster-role-to-user cluster-admin "system:serviceaccount:$($Env:TILLER_NAMESPACE):tiller"
oc adm policy add-scc-to-group anyuid system:authenticated
oc adm policy add-scc-to-user privileged "system:serviceaccount:$($Env:TILLER_NAMESPACE):tiller"
oc adm policy add-scc-to-user privileged system:serviceaccount:qlik:default
oc adm policy add-role-to-user admin system:serviceaccount:qlik:default
```
15. ストレージを定義し、次のコマンドを使用して書き込み可能にします。
```
minishift ssh -- "sudo chmod -R o+rwx /var/lib/minishift/base/openshift.local.pv*"
```
16. 次のコマンドを使用して、開発者としてログインします。
```
oc login -u developer
```
17. Qlik helm リポジトリを追加し、次のコマンドを使用して更新します。
```
helm repo add qlik https://qlik.bintray.com/stable
helm repo update
```
18. これで、minishift.yaml ファイルを使用して Qlik Sense Enterprise on Kubernetes を展開できるようになりました。
```
helm upgrade --install qsefe qlik/qsefe --set devMode.enabled=true,engine.acceptEULA="yes" -f C:\dev\minishift.yaml
```
19. 展開を監視するには、kubectl get pods または oc get pods を実行します。

4 インストール後の Qlik Sense Enterprise on Kubernetes の設定

このセクションでは、インストール後に Qlik Sense Enterprise on Kubernetes サイトを設定するプロセスについて説明します。組織特有の必要性に合うようサーバーを設定することができます。ほとんどの展開で必要となる一般的なタスクを以下に示します。

Qlik Sense Enterprise on Kubernetes の実装は、必要とされる構成に応じてさまざまです。以下のページでは、実稼働実装を行うために必要とされる主要な要素について説明しています。

- *Qlik Sense Enterprise on Kubernetes へのアプリの配布 (page 29)*
- *ID プロバイダーの設定 (page 29)*
- *Qlik Sense Enterprise on Kubernetes 展開での証明書の設定 (page 43)*
- *Qlik Sense Enterprise on Kubernetes での MongoDB の構成 (page 45)*

Qlik Sense Enterprise on Kubernetes のインストール時に、サービス間通信用の既定のサービスキー群が提供されます。実稼働に向けた保護の一環として、この既定のキー群を順番に変えながら使用することを推奨します。

4.1 Qlik Sense Enterprise on Kubernetes へのアプリの配布

Qlik Sense Enterprise on Kubernetes が実行されている場合は、Qlik Sense Enterprise on Windows 展開からそこにアプリを配布できます。アプリを配布するには、次のステップを実行します。

1. Qlik Sense Enterprise on Windows 展開の Multi-cloud Setup Console で展開を作成します。
「*Multi-Cloud セットアップコンソール - スタートページ*」および「*MSC - 展開*」を参照してください。
2. 配布するアプリケーションを決定するための配布ポリシーを作成します。
「*配布ポリシー - はじめに*」を参照してください。
3. アプリで、**collection**、**userswithaccess**、または **groupswithaccess** プロパティを設定して、アプリケーションを公開します。
「*クラウドハブのコレクションへのアプリの公開*」を参照してください。

アプリを配布すれば、hub からそれらのアプリを開くことができるようになります。

4.2 ID プロバイダーの設定

ID プロバイダー (IdP) は、ユーザーの ID 情報を管理し、認証サービスを提供します。ID プロバイダーは、繰り返しログインし直さなくても別の Web サイトにアクセスできるよう、single sign-on (SSO) を有効化します。Active Directory や LDAP などのオンプレミス技術と比べ、ID プロバイダーはクラウドサービスへのアクセス時に一貫性のある管理された体験も提供し、新しいサービスごとにアカウントを作成する必要がなくなります。

ユーザー アカウントが Active Directory に格納されている場合でも、IdP はクラウド ソフトウェアへの統合を有効化できます。

Qlik Sense Enterprise on Cloud Services、Qlik Sense Enterprise on Kubernetes、またはマルチクラウド展開では、IdP は以下を提供します。

- セキュアなユーザー認証、およびすべての展開間で受け渡される共通の ID (ユーザー ID とグループ)。
- ライセンス割り当てのための共通ユーザー ID (重複使用回避のため)。
- コンテンツへのアクセス制御を適用する際に使用する、共通のユーザー ID と属性 (グループなど)。

例: マルチクラウド展開における IdP

IdP の要件

Qlik Cloud Services と Qlik Sense Enterprise on Kubernetes の両方を OpenID Connect (OIDC) 標準を使用 IdP と統合します。これは、ユーザーがブラウザ経由でログインするインタラクティブ ログインと、ソフトウェア製品経由で API を使用する自動ログインとの両方を可能にする標準です。

Qlik Sense Enterprise on Windows は現在、OIDC に対応していませんが、SAML、すなわち IdP が提供する認証情報として一貫したユーザー IDを実現する手法に対応しています。

要約すると、マルチクラウドにおける IdP は OIDC と SAML の両方に対応する必要があります。

Qlik Sense Enterprise on Kubernetes をセットアップして使用できるようにするには、IdP から以下の情報を取得する必要があります。

- **discoveryUrl**: Qlik Sense などのアプリケーションが IdP を最小構成で使用できるようにする OpenID Connect Discovery URL。
- **clientId**: IdP からのクライアントを一意に識別。
- **clientSecret**: IdP での認証のためにクライアントがクライアント ID とともに使用するシークレット。
- **realm**: IdP に関連付ける名前。
- **hostname**: Qlik Sense Enterprise on Kubernetes の展開で使用されるホスト名。

これらの値は、Qlik Sense Enterprise on Kubernetes のインストール時に、**values.yaml** ファイルの identity-providers セクションの下に追加されます。

次の IdP ベンダーについて、この構成の例を段階的に示します。

- Okta
  「*Okta の設定*」を参照してください。
- Auth0
  「*Auth0 の設定*」を参照してください。

- ADFS
「*ADFS の設定*」を参照してください。

ADFS の設定

ADFS は認証および承認を行うプラットフォームです。

ADFS は、Qlik Sense Enterprise on Kubernetes (QSEoK) および Qlik Sense Enterprise on Windows (QSEfW) で使用する ID プロバイダー (IdP) として構成できます。アプリケーション グループ、サーバー アプリケーション、および Web API を作成し、インタラクティブ ログイン (QSEoK) に使用します。さらに、Active Directory から ID トークンにクレームをマッピングします。

インタラクティブ ログインのために QSEoK で必要とされる ADFS リソースの作成

ADFS を設定する場合、アプリケーション グループとサーバー アプリケーションが必要です。

> *以下の手順は、ADFS 10 を使用する場合の例です。詳細および最新の手順については、ADFS のドキュメントを参照してください。*

アプリケーション グループの追加とサーバー アプリケーションの作成

次の手順を実行します。

1. **[Add Application Group Wizard]** (アプリケーション グループの追加ウィザード) を開きます。
2. アプリケーション グループの名前を入力します。
3. **[Templat]** (eテンプレート) で、**[Server application]** (サーバー アプリケーション) を選択します。
4. **[次へ]** をクリックします。
[Server application] (サーバー アプリケーション) ページが開きます。
5. アプリケーションの名前を入力します。
例: *1234567890*
6. アプリケーションのクライアント識別子を入力し、書き留めておきます。クライアント識別子はクライアント ID として使用されます。
例: *https://adfs.elastic.example/1234567890*

   > *この例では、https://adfs.elastic.example がテナント ドメインで、1234567890 がアプリケーションを表す一意の識別子です。クライアント識別子は URL でなければなりません。ADFS は、URL ID によってアプリケーションを表す id_token 内にカスタム クレームのみを含みます。詳細については、「Customize claims to be emitted in id_token when using OpenID Connect or OAuth with AD FS 2016」を参照してください。*

7. **Redirect URI** については、*https://<host>/login/callback/* の書式で、テナントのログイン コールバックへのリダイレクト URL を設定します。
例: *https://adfs.elastic.example/login/callback*
8. 必要に応じて、説明を入力します。
9. **[次へ]** をクリックします。

[**Configure Application Credentials**] (アプリケーションの資格情報の構成) ページが開きます。

10. [**Generate a shared secret**] (共有シークレットを生成する) を選択します。このシークレットを書き留めておかないと、後でアクセスできなくなります。このシークレットは、クライアントシークレットとして使用します。
11. ウィザードを終了します。

アプリケーション グループへの Web API の追加

作成したアプリケーション グループに Web API を追加します。

次の手順を実行します。

1. あらかじめ作成したアプリケーション グループを開きます。
2. [**Add application**] (アプリケーションの追加) > [**Web API**] (Web API) の順に選択します。
3. アプリケーション グループから識別子としてクライアント ID を追加します。
4. [**次へ**] をクリックします。
   [**Choose Access Control Policy**] (アクセス制御ポリシーの選択) ページが開きます。
5. ポリシーを適用し、[**次へ**] をクリックします。
   [**Configure Application Permissions**] (アプリケーションの権限の構成) ページが開きます。
6. [**Permitted scopes**] (許可されているスコープ) について、*allatclaims*、*email*、*openid*、および *profile* を選択します。
7. ウィザードを終了します。

id_token に対するクレームの構成

次の手順を実行します。

1. 作成した Web API を編集するアプリケーション グループを開きます。[**Issuance Transform Rules**] (発行変換規則) を開きます。
2. ルール テンプレートの [**Send LDAP Attributes as Claims**] (LDAP 属性を要求として送信) を元にルールを作成します。
3. [**Active Directory**] を属性ストアとして選択します。
4. クレーム マッピングを追加します。必要に応じて、送信クレームを入力します。
5. [*Token-Groups - Unqualified Names*] (Token-Groups - 名前の指定なし) を [*groups*] (グループ) にマッピングします。
6. [*表示名*] を [*display_name*] にマッピングします。
7. クレームのマッピングを終了します。

Qlik Sense Enterprise on Kubernetes での IdP としての ADFS の使用

ADFS は、ADFS のユーザーを使用して Qlik Sense Enterprise on Kubernetes テナントにログインする場合の ID プロバイダーとして使用できます。

ADFS による Qlik Sense Enterprise on Kubernetes の接続

はじめに、以下が揃っていることを確認します。

- ADFS インストール
- ADFS 内で構成された必須リソース
- ADFS アプリケーションの構成設定 (*discoveryUrl*、*clientId*、および*clientSecret*)
- ハイブリッドデプロイヤーの値 (パブリック キー、キー ID、発行元)

コード例の多くはプレースホルダー値を含み、ユーザーの設定値で置き換える必要があります。

values.yml ファイルを使用して Qlik Sense Enterprise on Kubernetes に構成情報を入力します。*values.yml* ファイルの例は次のとおりです。

```
devMode:
  enabled: true

engine:
  acceptEULA: "yes"

identity-providers:
  secrets:
    idpConfigs:
      - discoveryUrl: "https://adfs-host/adfs/.well-known/openid-configuration"
        clientId: "https://adfs.elastic.example/1234567890"
        clientSecret: "<client secret>"
        realm: "ADFS"
        hostname: "adfs.elastic.example"
        useClaimsFromIdToken: true
        claimsMapping:
          sub: ["sub", "appid"]
          client_id: "appid"
          name: "display_name"
      - issuerConfig:
          issuer: https://the-issuer
        primary: false
        realm: "ADFS"
        hostname: "adfs.elastic.example"
        staticKeys:
        - kid: "thekid"
          pem: |-
            -----BEGIN PUBLIC KEY-----
            MHYwEAYHKoZIzj0CAQYFK4EEACIDYgAEsMSxQjXxrvqoKSAREQXsr5Q7+/aetjEb
            OUHt8/Cf73WD56cb4QbHthALl5Ej4MUFOAL9imDmVQe58o9b1j5Zo16Rt1gjLDvd
            nqstc+PX4tyxqGadItJAOU3jka7jYghA
            -----END PUBLIC KEY-----
```

userClaimsFromIdToken フラグが true に設定されている点が重要です。このフラグは、userinfo に対するクエリの代わりに ID トークンからのクレームを使用するよう edge-auth に指示します。これは、ADFS が userinfo の応答ではごくわずかな情報しか返さず、ID トークンにほとんどの情報を含めるためです。

自ら設定する値は、*discoveryUrl*、*clientId*、*clientSecret*、*realm*、および *hostname* に挿入する必要があります。

クラスターへの構成の適用

Helm (https://helm.sh/参照) を使用し、以下に示すように *values.yml* ファイルの構成を Kubernetes クラスターに適用します。

```
$ helm upgrade \
  --install \
  qliksense qlik/qliksense \
  -f values.yml
```

構成が適用されたことを確認するには、`get values` コマンドを実行し、解決された構成を表示します。

```
$ helm get values qliksense

devMode:
  enabled: true
engine:
  acceptEULA: "yes"
identity-providers:
  secrets:
    idpConfigs:
      - discoveryUrl: "https://adfs-host/adfs/.well-known/openid-configuration"
        clientId: "https://adfs.elastic.example/1234567890"
        clientSecret: "<client secret>"
        realm: "ADFS"
        hostname: "adfs.elastic.example"
        useClaimsFromIdToken: true
        claimsMapping:
          sub: ["sub", "appid"]
          client_id: "appid"
          name: "display_name"
      - issuerConfig:
          issuer: https://the-issuer
        primary: false
        realm: "ADFS"
        hostname: "adfs.elastic.example"
        staticKeys:
        - kid: "thekid"
          pem: |-
            -----BEGIN PUBLIC KEY-----
            MHYwEAYHKoZIzj0CAQYFK4EEACIDYgAEsMSxQjXxrvqoKSAREQXsr5Q7+/aetjEb
            OUHt8/Cf73WD56cb4QbHthALl5Ej4MUFOAL9imDmVQe58o9b1j5Zo16Rt1gjLDvd
            nqstc+PX4tyxqGadItJAOU3jka7jYghA
            -----END PUBLIC KEY-----
```

ホストファイルの構成

このセクションは、DNS がない場合のみ関係します。

`<hostname>` を解決するには、以下を `/etc/hosts` ファイルに追加します。

```
127.0.0.1   <hostname>
::1         <hostname>
```

テナントへのログイン

ADFS 展開のユーザーがテナントにログインする設定が完了しました。ブラウザで *https://<tenant address>* にアクセスすると、ADFS ログイン ページにリダイレクトされます。ログインに成功すると、アプリの配布先のホーム ページが表示されます。

Auth0 の設定

Auth0 は認証および承認を行うプラットフォームです。

Auth0 は、Qlik Sense Enterprise on Kubernetes (QSEoK) および Qlik Sense Enterprise on Windows (QSEfW) で使用する ID プロバイダー (IdP) として構成できます。

QCS または QSEoK のインタラクティブ ログインに使用する Auth0 アプリケーションと接続の作成

Auth0 アプリケーションを作成し、そのアプリケーションを Auth0 データベース接続に接続します。

Auth0 アプリケーションにより、アプリケーション (QSEfW/QCS/QSEoK) では認証に Auth0 を使用することができます。Auth0 接続はユーザーのソースであり、この例ではユーザーが追加されているデータベースです。

Auth0 アカウントとテナントは作成済みであると仮定します。

> *以下の手順は具体例です。詳細および最新の手順については、Auth0 のドキュメントを参照してください。*

Auth0 における新規アプリケーションの作成

次の手順を実行します。

1. Auth0 の左メニューで、**[Applications]** (アプリケーション) を開きます。
2. **[Create application]** (アプリケーションの作成) をクリックします。
3. アプリケーションの名前を指定し、**[Single Page Web Applications]** (シングル ページ Web アプリケーション) を選択して、**[作成]** をクリックします。
4. 必要に応じて、Web アプリテクノロジーを選択します。
5. **[設定]** を選択します。
6. **[Allowed Callback URLs]** (許可されたコールバック URL) ボックスで、*https://<host>/login/callback/* の書式でホストに URL を追加します。
7. 下にスクロールし、**[Save changes]** (変更を保存) をクリックします。
8. **[クライアント ID]** の値を書き留めます。
9. **[クライアント シークレット]** の値を書き留めます。
10. 一番下までスクロールし、**[詳細設定]** を選択します。
11. **[Endpoints]** (エンドポイント) タブを選択します。
12. 後で使用するため、**[OpenID configuration]** (OpenID 構成) の URL を書き留めておきます。

Auth0 でのデータベース接続の作成

データベース接続を作成し、この接続を使用するようアプリケーションを構成します。

次の手順を実行します。

1. 左メニューで、**[接続] > [データベース]** の順に選択します。
2. データベース接続の名前を入力し、**[作成]** をクリックします。
3. 左メニューで、**[Applications]** (アプリケーション) を選択します。
4. **[Connections]** (接続) タブを開きます。
5. アプリケーション用の新規データベース接続を有効にします。

新規ユーザーの作成 (任意)

次の手順を実行します。

1. 左メニューで、**[ユーザー]** を選択します。
2. **[Create your first user]** (最初のユーザーを作成) をクリックします。
3. 各項目を入力し、新規に作成した接続を選択します。

プログラム アクセスのための Auth0 API およびアプリケーションの作成

はじめに API を作成します。

プログラム アクセスを設定し、コンテンツを Qlik Cloud Services (QCS) または QSEoK に配布できるようにします。

Auth0 で新しい API を作成します。この場合、Auth0 API は保護された QSEoK リソース API を表します。OAuth の点から、Client Credentials Grant フローについて Auth0 を構成します。

はじめに、アプリケーションのための新規 API を作成します。

次の手順を実行します。

1. 左メニューで、**[APIs]** (API) を選択します。
2. **[Create API]** (API を作成) をクリックします。
3. API の名前を入力します。
4. **[Identifier]** (識別子) に「`qlik.api`」と入力します。
5. **[作成]** をクリックします。
6. **[Scopes]** (スコープ) タブに移動します。
7. 名前と説明の値に [any] (任意) を使用して新しいスコープを追加し、**[追加]** をクリックします。

上記のインタラクティブ ログイン用の Auth0 アプリケーションを作成したのと同様に、プログラム認証のための Auth0 アプリケーションを作成します。

次の手順を実行します。

1. 左メニューで、**[Applications]** (アプリケーション) を選択します。
2. **[Create Application]** (アプリケーションの作成) をクリックします。
3. **[Machine to Machine Applications]** (マシン間 アプリケーション) を選択します。
4. **[作成]** をクリックします。
5. 上記で作成した API を選択します。

6. [**Scopes**] (スコープ) ボックスで、[**any**] (任意) を選択します。
7. [**Authorize**] (承認) をクリックします。
8. [**Settings**] (設定) タブを選択します。[**Allowed Web Origins**] (許可された Web オリジン) ボックスで、展開に URL を追加します。
9. [**クライアント ID**] の値を書き留めます。
10. [**クライアント シークレット**] の値を書き留めます。
11. 一番下までスクロールし、[**詳細設定**] を選択します。
12. [**Endpoints**] (エンドポイント) タブをクリックします。
13. [**OAuth Token URL**] (OAuth トークン URL) の値を書き留めます。
この値は、クライアント ID およびクライアント シークレットとともに、展開を追加する場合に QSE for Windows の構成で使用されます。
14. 左メニューで、[**APIs**] (API) を選択し、新たに作成した API を開きます。[**Machine to Machine Applications**] (マシン間アプリケーション) タブを選択します。
15. 新しいアプリケーションに新しい Auth0 API へのアクセス権があることを確認します。

Qlik Sense Enterprise on Kubernetes での IdP としての Auth0 の使用

Auth0 は、Qlik Sense Enterprise on Kubernetes (QSEoK) テナントへのログインと、そのテナントとのプログラムによるインタラクティブなやり取りのための ID プロバイダーとして使用できます。

QSEoK と Auth0 の接続

はじめに、以下が揃っていることを確認します。

- Auth0 アカウント
- Auth0 テナント
- Auth0 アプリ。インタラクティブ ログインおよびプログラム アクセスについて構成済み
- Auth0 アプリケーションの構成設定: *discoveryUrl*、*clientId*、および *clientSecret*

コード例の多くはプレースホルダー値を含み、ユーザーの設定値で置き換える必要があります。

values.yml ファイルを使用して QSEoK に構成情報を入力します。*values.yml* ファイルの例は次のとおりです。

```
devMode:
  enabled: true

engine:
  acceptEULA: "yes"

identity-providers:
  secrets:
    idpConfigs:
      - discoveryUrl: "<OpenID Configuration from Application>"
        clientId: "<Client ID from Application>"
        clientSecret : "<Client Secret from Application>"
        realm: "<Name for this IdP>"
        hostname: "<Hostname for your QSEoK tenant>"
        claimsMapping:
```

```
        client_id: [ "client_id", "<id>" ]
```

discoveryUrl、*clientId*、*clientSecret*、*realm*、*hostname*、および *id* (クレーム マッピング) の値を入力する必要があります。

クラスターへの構成の適用

Helm (https://helm.sh/参照) を使用し、以下に示すように *values.yml* ファイルの構成を Kubernetes クラスターに適用します。

```
$ helm upgrade \
  --install \
  qliksense qlik/qliksense \
  -f values.yml
```

構成が適用されたことを確認するには、`get values` コマンドを実行し、解決された構成を表示します。

```
$ helm get values qliksense
```

```
devMode:
  enabled: true
engine:
  acceptEULA: "yes"
identity-providers:
  secrets:
    idpConfigs:
      - discoveryUrl: "https://tenant.auth0.com/.well-known/openid-configuration"
        clientId: "<client ID>"
        clientSecret : "<client secret>"
        realm: "Auth0"
        hostname: "<hostname>"
```

ホストファイルの構成

このセクションは、DNS がない場合のみ関係します。

`<hostname>` を解決するには、以下を `/etc/hosts` ファイルに追加します。

```
127.0.0.1   <hostname>
::1         <hostname>
```

テナントへのログイン

テナントにログインする準備ができました。ブラウザで *https://<tenant address>* にアクセスすると、Auth0 ログイン ページにリダイレクトされます。ログインに成功すると、アプリの配布先のホーム ページが表示されます。

Okta の設定

Okta は認証および承認を行うプラットフォームです。

このトピックでは、Qlik Sense Enterprise on Kubernetes (QSEoK) および Qlik Sense Enterprise on Windows (QSEfW) とともに使用する Okta の設定方法を説明します。Okta は、QSEoK および QSEfW で使用する ID プロバイダー (IdP) として構成できます。

以下を作成します。

- インタラクティブ ログインのためのアプリケーション (QSEoK)
- Okta のプログラムによる使用

インタラクティブ ログインのための Okta アプリケーションとQSEoK のユーザーの作成

Okta アプリケーションおよびユーザーを作成します。Okta アプリケーションにより、アプリケーション (QSEfW/Qlik Cloud Services (QCS)/QSEoK) では認証に Okta を使用することができます。

Okta アカウントとテナントは作成済みであると仮定します。

> *Qlik Sense Enterprise on Windows, with Multi-Cloud をインストールする場合、Okta の開発者アカウントを使用する必要があります (「Okta Developer」(Okta 開発者) 参照)*

ユーザーの作成

Okta でユーザーを作成します。すでにユーザーを作成済みの場合はこの手順を省略できます。

次の手順を実行します。

1. 姓および名を入力します。
2. **ユーザー名**: メール アドレスをユーザー名として使用します。
3. Primary email (プライマリ メール): [**ユーザー名**] と同じです。
4. [**パスワード**] は、[**Set by admin**] (管理者が設定) を選択します。
5. 新規ユーザーのパスワードを入力します。
6. 必要に応じて、[**User must change password in first login**] (初回ログイン時にユーザーがパスワードを変更) の選択を解除します。

Okta における新規アプリケーションの作成

Okta から新しいアプリケーション、QSEoK 向けのテナントを作成します。

次の手順を実行します。

1. Okta で、[**Applications**] (アプリケーション) に進み [**Add Application**] (アプリケーションの追加) をクリックします。
2. [**Platform**] (プラットフォーム) で、[**Web**] を選択し、[**次へ**] をクリックします。
3. アプリの名前を入力します。
4. ベース URI を入力します。

   > *これは、QSEoK から取得した IP アドレスまたはサーバー名です。例: https://40.118.9.61*

5. ログイン リダイレクト URI を入力します。
   ベース URI については、使用環境の IP アドレスまたはサーバー名を使用します。例: *https://40.118.9.61/login/callback*
6. [**Grant type allowed**] (許可される付与タイプ) セクションで、クライアント自身の代わりとなるクライアント

について、[**Client Credentials**] (クライアント資格情報) を選択します。

7. [**完了**] をクリックします。

プログラム アクセスに関する構成

プログラムによる使用をサポートするようOkta を構成します (この場合は、QSEoK または QCS に対する配布に対応するため)。

プログラム アクセスのための Okta API リソース サーバーおよびアプリケーションの作成

Okta で、新規の Resource Server API を作成します。この場合、Okta Resource Server API は、保護された QSEoK リソース API を表します。OAuth の点から、Client Credentials Grant フローについて Okta を構成します。

はじめに、テナントについて新規の Authorization Server を作成します([API] タブで)。

次の手順を実行します。

1. トップ メニューで、[**API**] を選択します。
2. [**Authorization Servers**] (承認サーバー) を開きます。
3. [**Add Authorization Server**] (承認サーバーの追加) をクリックします。
4. 名前、対象 (必ず `qlik.api`) および説明を入力します。
5. API を保存します。
6. [**Scopes**] (スコープ) タブを開きます。
7. [**Add Scope**] (スコープの追加) タブをクリックします。
8. 名前と説明を入力し、[**Set as default scope**] (既定のスコープとして設定) を選択します。
9. [**作成**] をクリックします。
10. [**Access Policies**] (アクセス ポリシー) タブを開きます。
11. [**Add Policy**] (ポリシーの追加) をクリックします。
12. 名前と説明を入力し、[*Grant Clients*] (許可するクライアント) を入力します。
13. [**Assign to**] (割り当て先) は、[**All clients**] (すべてのクライアント) を選択したままにします。
14. [**Create Policy**] (ポリシーの作成) をクリックします。
15. [**Add Rule**] (規則の追加) をクリックします。
16. ルールの名前を入力します。
17. [**Client acting on behalf of a user**] (ユーザーの代わりとなるクライアント) の選択を解除します。
18. [**Create Rule**] (ルールの作成) をクリックします。

プログラム認証のための Okta アプリケーションの作成

上記のインタラクティブ ログイン用の Okta アプリケーションを作成したのと同様に、プログラム認証のための Okta アプリケーションを作成します。

次の手順を実行します。

1. Okta トップ メニューで、[**Applications**] (アプリケーション) を開きます。
2. [**Add Application**] (アプリケーションの追加) をクリックします。

3. [**Platform**] (プラットフォーム) で、[**サービス**] を選択し、[**次へ**] をクリックします。
4. アプリの名前を入力します。
5. [**完了**] をクリックします。

Qlik Sense Enterprise on Kubernetes での IdP としての Okta の使用

Qlik Sense Enterprise on Kubernetes (QSEoK) は、ID プロバイダーとして Okta を使用するように構成できます。

手順が完了すれば、Okta ユーザー名とパスワードを使用して QSEoK テナントにログインするとともに、QSEoK テナントをプログラムによりインタラクティブに操作することができるようになります。

ここでは、Docker for Mac を使用して Kubernetes を実行している Mac 上で QSEoK が実行されていると仮定します。これと同じ構成でなくても、対応する別の方法で Kubernetes を実行していれば、同じコンセプトを利用することができます。

Okta IdP を使用するための QSEoK の設定

はじめに、以下が揃っていることを確認します。

- Okta アカウント
- Okta テナント
- Okta アプリ。インタラクティブ ログインおよびプログラム アクセスについて構成済み。
- Okta アプリケーションの構成設定:
  - *discoveryUrl*:QSEoK などのアプリケーションで Okta を最小構成で使用できるようにする OpenID Connect Discovery URL。
  - *clientId*: 認証に Okta を使用するクライアントを一意に識別します。
  - *clientSecret*: 認証に Okta を使用するため、クライアントがクライアント ID とともに使用するシークレット。

コード例の多くはプレースホルダー値を含み、ユーザーの設定値で置き換える必要があります。

values.yml ファイルを使用して QSEoK に構成情報を入力します。*values.yml* ファイルの例は次のとおりです。

```
devMode:
  enabled: true

engine:
  acceptEULA: "yes"

identity-providers:
  secrets:
    idpConfigs:
      - discoveryUrl: "<OpenID Configuration from Application>"
        clientId: "<Client ID from Application>"
        clientSecret : "<Client Secret from Application>"
        realm: "<Name for this IdP>"
        hostname: "<Hostname for your QSEoK tenant>"
```

discoveryUrl、*clientId*、*clientSecret*、*realm*、および *hostname* の値を入力する必要があります。

Okta では、作成したアプリケーションの [**Client Credentials**] (クライアント資格情報) セクションにある [**基本設定**] タブの下で、[*クライアント ID*] と [*クライアント シークレット*] を確認できます。

クラスターへの構成の適用

Helm (https://helm.sh/ 参照) を使用し、*values.yml* ファイル内の構成を Kubernetes クラスターに適用します。

```
$ helm upgrade qliksense qlik/qliksense -f values.yml
```

構成が適用されたことを確認するには、次に示す `get values` コマンドを実行し、解決された構成を表示します。

```
$ helm get values qliksense
devMode:
  enabled: true
engine:
  acceptEULA: "yes"
identity-providers:
  secrets:
    idpConfigs:
      - discoveryUrl: "https://dev-<tenantid>.oktapreview.com/.well-known/openid-configuration"
        clientId: "<clientID code>"
        clientSecret : "<clientsecret code>"
        realm: "Okta"
        hostname: "<hostname>"
```

ホストファイルを構成しています

このセクションは、DNS がない場合のみ関係します。

`<hostname>` を解決するには、以下を `/etc/hosts` ファイルに追加します。

```
127.0.0.1   <hostname>
::1         <hostname>
```

テナントへのログイン

テナントにログインする準備ができました。ブラウザで *https://<tenant address>* にアクセスすると、Okta ログインページにリダイレクトされます。ログインに成功すると、アプリの配布先のホームページが表示されます。

QSEoK へのプログラムによる構成の追加

ここでは、上記で作成したアプリケーションおよび認証サーバーを指定するために、QSEoK に対する IdP 構成が必要になります。`primary: true` が既存の構成に追加されていた点に注目してください。

```
devMode:
  enabled: true

engine:
  acceptEULA: "yes"

identity-providers:
  secrets:
    idpConfigs:
      - discoveryUrl: "https://dev-<tenantid>.oktapreview.com/.well-known/openid-configuration"
        clientId: "<client ID coder"
        clientSecret : "<client secret code>"
        realm: "Okta"
```

```
        hostname: "<hostname>"
        primary: true
      - discoveryUrl: "https://dev-<tenantid>.oktapreview.com/oauth2/<resource-server-id>/.well-
known/openid-configuration"
        primary: false
        realm: "Okta"
        hostname: "<hostname>"
        claimsMapping:
          client_id: ["client_id", "cid"]
```

Helm を使用し、*values.yml* ファイル内の構成を Kubernetes クラスターに適用します。

```
$ helm upgrade qliksense qlik/qliksense -f values2.yml
```

4.3 Qlik Sense Enterprise on Kubernetes 展開での証明書の設定

既定では、ユーザーのブラウザーでは信用されない自己署名証明書を使用して Qlik Sense Enterprise on Kubernetes がインストールされます。この証明書を自分が所有する SSL 証明書と置き換えるには、以下のステップを実行します。

> *この例では、証明書が tls.crt というファイルに、また関連付けられたプライベートキーが tls.key というファイルに保存されています。*

Kubernetes でシークレットリソースを作成する

1. 証明書と証明書のキーを格納するための secret.yaml というファイルを作成します。一例として、以下の yaml の定義を参照してください。

```
apiVersion: v1
kind: Secret
metadata:
  name: my-certificate
  namespace: default
type: kubernetes.io/tls
data:
  tls.crt:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURORENDQWh3Q0NRRHUxeDdVdEdJSjJEQU5CZ2txaGtpRzl3MEJBUX
NGQURCY01Rc3dDUVlEVlFRR0V3SkQKUVRFUU1BNEdBMVVFQ0F3SFQyNTBZWEpwYnpFUE1BMEdBMVVFQnd3R1QzUjBZWGRo
TVJRd0VnWURWUVFLREF0TQpaWGhqYjNKd0lFbHVZekVVTUJJR0ExVUVBd3dMYkdWNFkyOXljQzVqYjIwd0hoY05NVGd3Tm
pJM01Ua3hOelV3CldoY05NVGt3TmpJM01Ua3hOelV3V2pCY01Rc3dDUVlEVlFRR0V3SkRRVEVRTUE0R0ExVUVDQXdIVDI1
MFlYSnAKYnpFUE1BMEdBMVVFQnd3R1QzUjBZWGRoTVJRd0VnWURWUVFLREF0TVpYaGpiM0p3SUVsdVl6RVVNQklHQTFVRQ
pBd3dMYkdWNFkyOXljQzVqYjIwd2dnRWlNQTBHQ1NxR1NJYjNEUUVCQVFVQUE0SUJEd0F3Z2dFSOFvSUJBUUQyCndTZk03
TDJiTHBnR1VsRERIci8rSUc3YlFONndnTzVMYWdqcnJramFHRjRGcVc0NS9Ha3hHTVhlZzZSTUpuYnkKTTI1RVloY2ZSVl
lzTmtaQVpCakYzM2ZwNlBqYjhydzhzV016RnJMOUtkeG8rZEtYSE14MTkvaExTaS82QTJOMgpBNzJta1krT2JmMHlOR1B5
aEZVYOlEZEFxbWtGTitoTXlGZjQwSOl5VS94NjZMVHhsYjZLQm1uZm9LK3VlNStZCmVxcGVLRkhBZkZwK1NFSG5UMkNJZX
dmQXR0Z29NL3dyREZVcENPS0sxZEJMUytzbzBZOUFCWG9wRm05U1RGV00KNVdZMno1NWdoa1l6UUpmemlRMC9WZkppamdM
WmhwcjRCRVAwaXV0dlRIczhrSFIyUTJqSkxVNEpncFViTnlXOQpUejFLQzhMVGpiL1J4RVJCK01FWkFnTUJBQUV3RFFZSk
tvWklodmNOQVFFTEJRQURnZ0VCQUlJSitsc2tlcUwzClpPVmxwNm1ZakRmRGt5Z1dkQVlCU0pINDdidHJkQUZrSlpQVVVM
```

Tzl6cTAxd0VBUHArRzFqZGUzSG42QmtyM0cKcklsRFZEckFobGRLR0hONS9BMXdoMjBxVlNKZzBkV25ielJnSVlkK0ljdj
hIa1RORGlUSDRxWHRjUHR2VHQ0NgphTmJUOHc2THl0Sm9YRWNZVEQzcjVXdERKWHFodkdHUjVLNU1Ubmo3QmZxcXNsS2M4
ZUZVbFdxOUpJZEZGZGRNClNzbHlhdE1zbk9YMGxtdDc4VnZYRFJ1RU5RM3BYUU1wNkhyQ3ZHRnY0NlhLRE5scVNyZm9vSU
RvbHBaUW5CdHAKWmdGZHc4U3VtMW91NVo1V0QxeW9XYkdLdFlLaTFiZkdNRHlsNURqZHdIakk0dG5GZk5LQ2E5TGZMdG5h
S2V2RwprcS8wOTBtcjJxcz0KLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQo=
tls.key:
LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcFFJQkFBS0NBUUVBOXNFbnpPeTlteTZZQmxKUXd4Ni8vaU
J1MjBEZXNJRHVTMm9JNjY1STJoaGVCYWx1Ck9meHBNUmpGM29Pa1RDWjI4ak51UkdJWEgwVldMRFpHUUdRWXhkOTM2ZWo0
Mi9LOFBHRmpNeGF5L1NuY2FQblMKbHh6TWRmZjRTMG92K2dOamRnTzlwcEdQam0zOU1yUmo4b1JWSENBM1FLcHBCVGZvVE
1oWCtOQ2lNbFA4ZXVpMAo4WlcraWdacDM2Q3ZybnVmbUhxcVhpaFJ3SHhhZmtoQjUwOWdpSHNId0xiWUtEUDhLd3hWS1Fq
aWl0WFFTMHZyCktOR1BRQVY2S1JadlVreFZqT1ZtTnMrZVlJWkdNMENYODRrTlAxWHlZbzRDMllhYStBUkQ5SXJyYjB4N1
BKQjAKZGtOb3lTMU9DWUtWR3pjbHZVODlTZ3ZDMDQyLzBjUkVRZmpCR1FJREFRQUJBb0lCQVFEaW9FNU1vTllYazNpZQp2
YlZkTDMzSUNjT203WEpaaTJEUXRLZFN4alEwMHBKd0FzZXd6QWxVeFZyZDNldms0S0w0R1pKWmpmbU1oK0w0CklqVHRhTU
Z6NHFWQW1POFBHMllVMHFFSVIvM0dGRTlSdnJqU2Z1bXVJenZROG1jVDZVN05FZXg1OGxCMTBNRHUKYzgwajdMUTZhOEF6
VFErOWNqYVJacU9kYXdpOHpLWnVpdkhRazFXelU0eTZyYUhRSEliVnZOWHAxMEx0K1RKYQpsRk83RlVwOG05bGh5UllqSy
9Vczd6Y0JOaWVRSWRTVGV6QkUvNzlwY0d4Vmd0Yk4xYTkyS2FjR0YzQWpzODF4CnNSQkc2Z2dMbTJ4U25Qdng2Y1ZvRkw4
VVNERUtOcDJBKzNnZlh0ZWg4RWNQSDNVcEZNT29YK0swRkh5bUxnY08KZ1g2R0U4OGRBb0dCQVA1MDZaRlcrTTRpbG1ibk
42VEFoeVpjbXY3RW1oVDJDb3NaeGUwbUYwcGE4YVUzWmhoOAorUk5nYk4rTjAwbWU3cmtmd2F6L2pMZUpxYUhibTdFSXdE
NWVQakRsR2lRMmh3L0ZPdzhCWkc5OFYzZVJBblQzCmE3UWxCVmRJcGRUaDVHYjlPbDhFbmN0Ti9ncXdaRi9Xa2wwSmF5VX
pHdzg0elFsMm4vRWg1MitqQW9HQkFQaEEKU0wzL3NCK3U2aDkzSUdBOFR4UURPQjJXMnFpY1g4aXkrd3p2U056Z1BCRTlT
ZTFSQytCMHNPZnV5NStGYUw1OApuazVPSnVjeWVuNXlkYTB1dndaV0ZFUWtkVGxtclRIV3F1d1ZCWWpad1RzVHBuTFp6TW
g1ZWE2TTEzV3ZRamdOClRGK1pENUd1QUZuaEpkaHlFbGg5NllpVXFvZkhCbVByYlVsMzNpZ1RBb0dCQU9scXFjOXZMckU1
UFNxTU01am8KRzVIdkpTZExoZVJ3aHQ5dTl3ZENCOTluYVgrYlFBWjZyYWsrck9DdG93Skh6c2oyL3AwSmx6WE9ERk96dG
pCQgp6TWZwZjdjdWtqcGg0cWR1ejJ4R1pMRjRLR3hBRXpia3VHSDZDOW96aEJ1eVUraTlwa3YvV2hoakpRVDlKalpjCkNN
dGJsblA3VzNrdEs3amdubnQrRkdWdEFvR0JBTHk5RXhEdzdsU0lSWk82bkRET2FVakwwY1FkUnd6ZUpxeU4KOXZTMGorN3
R4SDFPM1gvQ0dJZUQ4R1BGVjZabVpXWTlsSlh3TVAraGp5UEhuZFFYTUVCKy9WVjVhbTVEcy9XcQpRYlRQbnErVzdxUWRv
blM3UmtnOG90aWtWOE12aGViYnBXOGhEWWN5aFMzUVVWZW9FbTZZL0E1TkNRSEZ2UVFHCm44WTFqZjlsQW9HQUM2UFFYZ3
dwOTdudzMrWnZBeDJSUktqOHA2VUtraXVpRUlTQnM2ekR4c2hxdHJqR2VpaUQKRDNUNnBHVmhlYU94Y05reFRIVTlCcVo4
blNxd080cThjS2xLdXB0SkZFcFNtOVMrMWMveFdvNTdyWlVuNWlLbApDQzFWTkR6anYrRWNjU1FjOGJPZUxTRjd4cjRyYT
dnK3pRR0lSdXlvK005TDVBbjJxaEphM1ZRPQotLS0tLUVORCBSU0EgUFJJVkFURSBLRVktLS0tLQo=

2. name フィールドに分かりやすい名前を入力することができます。この例では、「my-certificate」を使用します。
   tls.crt フィールドは、証明書の base64 エンコード値です。この値は、以下のコマンドを使用して求めることができます。
   ```
   cat tls.crt | base64
   ```
3. base64 エンコード値が画面に表示されます。.yaml ファイルの tls.crt 値としてその値を入力します。
4. tls.key についても同じようにして値を求めます。
   ```
   cat tls.key | base64
   ```
5. .yaml ファイルに結果の base64 値を入力します。
6. ここで、以下のコマンドを使用して、Kubernetes でシークレット リソースを作成します。
   ```
   kubectl apply -f secret.yaml
   ```
7. シークレット リソースが作成されたことは、以下のコマンドを使用して確認できます。
   ```
   kubectl get secret my-certificate
   ```

証明書を使用するようにイングレスを設定する

1. 以下の行を values.yaml ファイルに追加することにより、前の手順で作成したシークレットを使用するように Qlik Sense イングレスを設定します。
   ```
   # References the “my-certificate” secret created within the “default” namespace
   elastic-infra:
    nginx-ingress:
   ```

```
controller:
  extraArgs:
    default-ssl-certificate: "default/my-certificate"
```

2. 次のコマンドを使用してクラスターを更新します。
   `helm upgrade --install qliksense qlik/qliksense -f values.yaml`

ブラウザーでの証明書の確認

- ブラウザーを使用して、設定したドメインに移動し、Qlik Sense のイングレス コントローラーによって提示された証明書を確認します。

4.4 Qlik Sense Enterprise on Kubernetes での MongoDB の構成

Qlik Sense Enterprise on Kubernetes は、いくつかのサービスの持続コンテンツ向け (Qlik Sense アプリファイルを除く) のデータベースとして **MongoDB** を使用します。

既定では、Qlik Sense Enterprise on Kubernetes (QSEoK) のインストール時に事前構成済みの **MongoDB Community Edition** が追加されます。これは、クイック スタート、テスト、評価の目的でのみ使用することを意図されています。このバージョンを使用する場合は、Kubernetes クラスターが更新されるとMongoDB のデータが失われる可能性があります。

実稼働可能な **MongoDB** 環境は次の方法でセットアップできます。

- 別々の MongoDB サーバーまたはクラスターを Qlik Sense と並列に展開します。
- MongoDB DBaaS プロバイダー (**MongoDB Atlas** または **mlab** など) を使用する

MongoDB 接続の構成

QSEoK のインストール時には、使用する MongoDB 接続を次のようにして指定できます。

- `helm install` コマンドのパラメータ。
- **values.yaml** の接続設定を参照し、その情報を `helm install` コマンドで使用する。

CLI パラメーターの使用

基本 `helm install` コマンドは、次のプロパティを設定することによって拡張できます。

- **devMode.enabled** 値を `false` に設定して開発モードを無効にします。
- 接続文字列を含む **mongodb.uri** 値を MongoDB に設定します。

Example:

```
helm upgrade \
    --install qliksense qlik/qliksense \
    --set mongodb.uri=<your-connection-string>,engine.acceptEULA="yes"
```

Example: 接続 URI のフォーマット

接続文字列のフォーマットは使用する構成によって変わります。Kubernetes の Secret 機能を使用して接続する場合、フォーマットは異なります。

```
mongodb://user:password@mongodbhost:port/databasename
```

> *一部のサービスでは、MongoDB の SSL が有効になっているとみなされています。SSL を使用していない場合は、接続 URI の末尾に `?ssl=false` を追加します。*

参照 values.yaml

values.yaml ファイルを作成し、`helm install` コマンドで参照する設定を含めます。

- **devMode.enabled** 値を `false` に設定して開発モードを無効にします。
- 接続文字列を含む **mongodb.uri** 値を MongoDB に設定します。

Example: values.yaml

```
engine:
  acceptEULA: "yes"

devMode:
  enabled: false
mongodb:
  uri: "<your-connection-string>"

identity-providers:
  secrets:
    idpConfigs:
      - <your IdP configuration here>
```

これで、values.yaml ファイルが `helm install` コマンドで参照されます。

```
helm upgrade \
  --install qliksense qlik/qliksense \
  -f values.yaml
```

SSL を使用した MongoDB への接続

MongoDB リポジトリデータベースへの、セキュア ソケット レイヤー (SSL) を使用した接続をセットアップできます。

> *Qlik Sense Enterprise on Kubernetes June 2019 およびそれ以降で動作します。*

MongoDB サーバーは SSL 証明書とともに設定され、SSL 接続が許可される必要があります。

接続 URI の末尾に `?ssl=true` を追加し、MongoDB リポジトリデータベースに SSL を使用して接続します。

```
devMode:
  enabled: false
```

```
mongodb:
  uri: <your-connection-string>?ssl=true
```

MongoDB データベースが自己署名証明書、またはパブリック認証局 (CA) により発行されたものではない証明書によりセットアップされた場合、CA 証明書チェーンを **values.yaml** ファイルに追加する必要があります。

```
global:
  certs:
    enabled: true
    configMap:
      create: true
      name: "{{ .Release.Name }}-ca-certs"
      certs: |+
        -----BEGIN CERTIFICATE-----
        MIIDLDCCAhSgAwIBAgIQANxWuceSqgA8h3fJ1Q7ZiTANBgkqhkiG9w0BAQsFADAm
        MSQwIgYDVQQDDBtRbGlrU2VydmVyMi5kb21haW4ubG9jYWwtQ0EwHhcNMTcwOTA5
        MTA1NjMwWhcNMjcwOTE2MTA1NjMwWjAmMSQwIgYDVQQDDBtRbGlrU2VydmVyMi5k
        b21haW4ubG9jYWwtQ0EwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCZ
        mnjqNBml1RU6vbRlakPNrCasFZSherijN4Rzj54BMb0cljC2ZfOnve2ZS5k27Dp9
        Yt/S30B6MQTRNZBCOCOw3jqnOW87iemxhoE713EkF+zNcWnhHRA53+be1iIhV+kE
        fyFl6/4QQmUmZo2hu6gIajmdtZvM/CgjiPdF5a6KQp4WHA339afuIMR5KQe1Qt7E
        xqaTbh7niOJEXZXSHcbT80sFam4O36rGMpjuseDbJgsIlLGSw1QwnIxWf+bFlbiD
        +2XJclAVXt+BCrsYfBXd3akLDulStW+X9SFFqX1V8+rkDdOV8lffaNhN6K4HQeBG
        ...
        LmIMzgUM9+baRYUwC552X6+szY55xqY21OyjFGSqrZDyyrJMi9RhDhSL1ZqIIJDm
        kpjNMY87Qa/c2slWTjg9lE/55OnBFZfQoD1zODVALCil9Tlb43wRsn8nMdD4U6Qz
        cgYfPkhRw2oUZuZwTmPOIYMrWPmmGXY4T9lZrq5afS7p+et1TKXZEZAC7akXDYL4
        CRjjXsfxmDaxy8sefg+LOnHgvESc1hWEBD2LlVVWbZCFi4MrwkkDyik5NWu6GkN2
        2xi+CJX3EBhHb1aFVDGd5dBSv3agXatsAnUMzxquuvtKbrURbMfyPCyiAZwlG9AN
        -----END CERTIFICATE-----
```

参考

- **enabled**: グローバル CA 証明書の使用を有効 (`true`) または無効 (`false`) にする。
- **create**: CA 証明書 configMap の作成を有効 (`true`) または無効 (`false`) にする。
- **name**: リリース名に基づく、テンプレート化された CA 証明書 configMap 名。
- **certs**: グローバル CA 証明書チェーン。これはあらゆる既存の CA トラストチェーンを置き換えます。

4.5 Qlik Sense Enterprise on Kubernetes での Qlik License Service 通信のプロキシの構成

Qlik License Service と License Back-end 間の通信は、プロキシを使用して処理できます。

Qlik License Service は Qlik Sense Enterprise February 2019 以降のリリースに含まれており、Qlik Sense が署名付きキー ライセンスを使用してアクティブ化されるときに使用されます。Qlik License Service は、製品アクティブ化と資格管理のために、Qlik にホストされている License Back-end Service に接続します。License Back-end Service へのアクセスとライセンス情報の取得には、ポート 443 が使用されます。

Qlik Sense June 2019 以降では、Qlik License Service と Qlik License Back-end 間の通信を、プロキシによって処理されるように構成できます。

Qlik Sense Enterprise on Kubernetes では、Qlik License Service のプロキシの構成は、helm 構成を使用して行います。HTTP および HTTPS の両方のスキームに対応しています。

[Licenses] (ライセンス) セクションの [*values.yaml*] ファイルに次の内容を追加します。

```
## Proxy configuration
## Set the following values when deploying behind a proxy
 proxy:
  ## The URI to the tunneling proxy scheme://host:port (e.g. http://proxy.company.com:8888)
  uri:
```