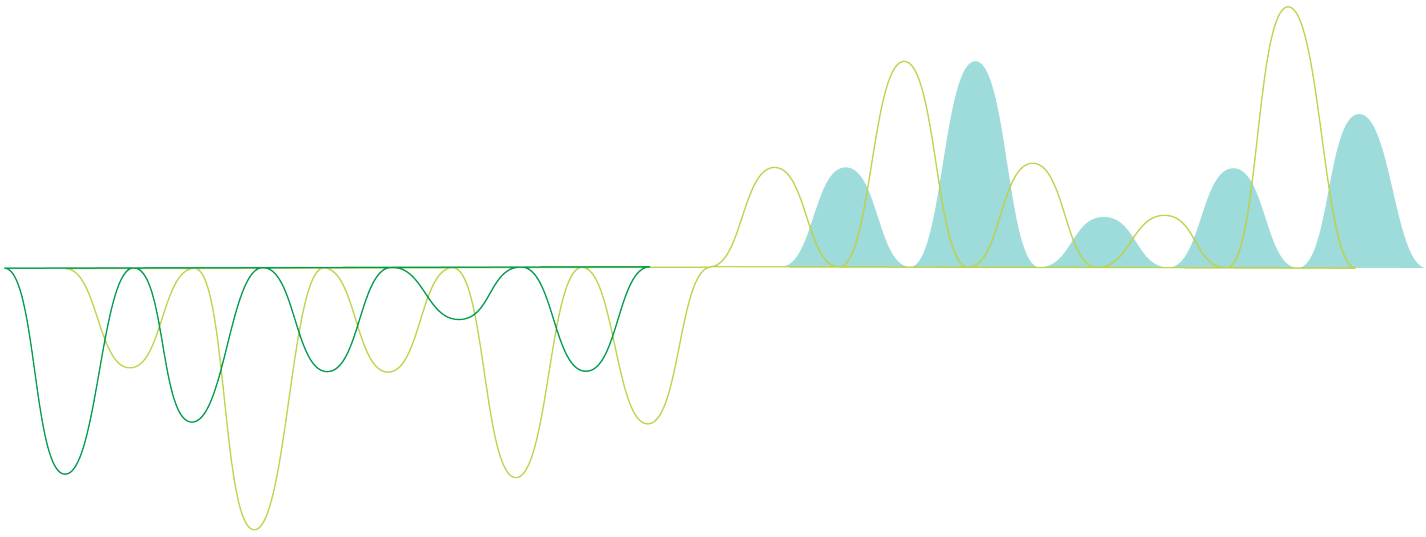
Deploy Qlik Sense Enterprise on Kubernetes

Qlik Sense®
November 2019



Contents

Contents

1 Planning your Qlik Sense Enterprise deployment

To successfully plan and prepare for your Qlik Sense deployment, do the following:

Introducing Qlik Sense Enterprise
: Get a brief introduction to Qlik Sense Enterprise.

Qlik Sense Enterprise deployment examples
: See different examples of deploying Qlik Sense Enterprise.

System requirements for Qlik Sense Enterprise
: Review the Qlik Sense Enterprise system requirements.

Qlik product licenses
: Understand how Qlik Sense uses license keys and LEF for site licensing.
Understand how Qlik Sense uses tokens for user access allocation (token-based licensing).
Ensure that you have your Qlik Sense license key available.

1.1 Qlik product licenses

Here is a summary of the license options that are available for the different Qlik Sense related products. Licensing allows you to manage the usage of the Qlik Sense software in your organization.

For detailed information on Qlik Sense licensing options, see Qlik's legal terms, product terms, and Licensing Service Reference Guide:

- Qlik Legal Terms
- Qlik Product Terms
- Qlik Licensing Service Reference Guide

Product activation

All Qlik products are entitled and enforced by a LEF (License Enabler File). The LEF is the artifact that is downloaded during the production activation. A Qlik product is licensed and activated using either a serial and control number, or a signed license key. The use of a signed license key is required for Qlik Sense Enterprise on Cloud Services and Qlik Sense Enterprise on Kubernetes deployments, and for the use of capacity based licenses.

With a signed license key, you need internet access (direct or through a proxy) to access the cloud-based license backend, for user assignments, analytic time consumption, and product activations.

Unified license

As of the April 2019 releases of Qlik Sense and QlikView, Qlik Sense customers can use a unified license in multiple deployments. A unified license shares the same signed key between:

- multiple Qlik Sense Enterprise deployments
- multiple QlikView Server deployments
- QlikView Server and Qlik Sense Enterprise deployments

Applying the same signed key to multiple deployments lets you share the same users and access types. Users can access all connected deployments using the same Professional or Analyzer access allocation.

Qlik Sense Enterprise

Qlik Sense Enterprise is the server version of Qlik Sense that you can deploy on a single node, or on multiple nodes. Qlik Sense Enterprise deployments are licensed and activated using a serial and control number, or a signed license key. Your Qlik Sense Enterprise license is based either on access types, or on tokens.

Access types

Access types licenses are the Professional and Analyzer Users licenses (user-based) and Analyzer Capacity licenses (capacity-based). You can combine these for a subscription based license if you use the signed license key when your deployment is activated. You can combine only user-based licenses if you are using a perpetual license.

After changing to a license with a signed key, you cannot return to using the old LEF based license model.

Token

You use tokens to allocate access passes to users so that they can access Qlik Sense. The License Enabler File (LEF) determines the number of tokens that you can allocate to different access passes. A user without an access pass cannot access apps.

With a Qlik Sense Token license you use tokens to allocate access passes to users. You can allocate user access and login access.

The token license is available only to customers with existing Qlik Sense Token licenses.

Core-based site

Qlik Sense Enterprise core-based sites are licensed based on the number of CPU cores on which the software will operate. A Core means a single processing unit within a processor or CPU, whether physical or virtual, including a vCPU or virtual core, which is capable of executing a single software thread at a time.

Qlik Sense Desktop

Qlik Sense Desktop is a free version of Qlik Sense that you can download on your computer, see www.qlik.com. To download Qlik Sense Desktop, you are required to create a Qlik account, and accept the Qlik Sense Desktop license agreement. In addition, when you install the software, you need to accept the software license agreement.

Qlik DataMarket

Qlik DataMarket offers a collection of up-to-date data from external sources accessible directly from within Qlik Sense. The available data includes current and historical weather and demographic data, currency exchange rates, as well as business, economic, and societal data.

Qlik DataMarket has two license options, one free and one licensed. The free option gives you access to a limited data set. The licensed option gives you access to premium data packages. You activate the license in the same way as for Qlik Sense Enterprise by entering owner information, serial number, and control number.

It is not necessary to accept Qlik DataMarket terms and conditions when using Qlik Sense Desktop. Access credentials are also not required because the premium data sets are not available on Qlik Sense Desktop.

Qlik NPrinting

You can install and configure Qlik NPrinting to connect to QlikView documents or Qlik Sense apps. The licensing requirements and procedures are different depending on if you connect Qlik NPrinting to QlikView or Qlik Sense.

Qlik NPrinting versions 16.0.0.0 and later are licensed by a LEF. If you are using an earlier version of Qlik NPrinting, we suggest that you upgrade to Qlik NPrinting versions 16.0.0.0 or later.

A Qlik Sense token is not required for the Qlik NPrinting service account. However, because you often perform troubleshooting within the Qlik NPrinting service account, it is helpful to assign a token to the Qlik NPrinting service account so that it has access to the Qlik Sense hub.

Qlik Sense Licenses

Qlik Sense Enterprise is the server version of Qlik Sense that you can deploy on a single node, or on multiple nodes. Qlik Sense Enterprise licenses are based either on access types, or on tokens.

For detailed information on Qlik Sense licensing options, read Qlik's legal terms, product terms, and Licensing Service Reference Guide:

- Qlik Legal Terms
- Qlik Product Terms
- Qlik Licensing Service Reference Guide

If you want to set up Qlik Cloud Services or Qlik Sense Enterprise on Kubernetes, please contact your Qlik representative or Qlik Support to obtain a valid license for the setup.

Unified license

As of the April 2019 releases of Qlik Sense and QlikView, Qlik Sense customers can use a unified license in multiple deployments. A unified license shares the same signed key between:

- multiple Qlik Sense Enterprise deployments
- multiple QlikView Server deployments
- QlikView Server and Qlik Sense Enterprise deployments

Applying the same signed key to multiple deployments lets you share the same users and access types. Users can access all connected deployments using the same Professional or Analyzer access allocation.

Qlik Sense Enterprise

A Qlik Sense Enterprise deployment can be licensed using two different models: the serial and control number and the signed license key. The License Enabler File (LEF) defines the terms of your license and the access types that you can allocate to users. Your Qlik Sense Enterprise license is based either on access types, or on tokens. A core-based license is also available. The use of a signed license key is required for Qlik Sense Enterprise on Cloud Services and Qlik Sense Enterprise on Kubernetes deployments, and for the use of capacity based licenses.

With a signed license key, you need internet access (direct or through a proxy) to access the cloud-based license backend, for user assignments, analytic time consumption, and product activations.

User-based and capacity-based licenses

A user-based license grants a predefined number of access allocations that can be assigned to unique and identified users. In Qlik Sense Enterprise, user-based licenses are either Professional and Analyzer Users licenses, or User access passes allocated with a Token license.

A capacity-based license grants a predefined number of time allocations for accessing Qlik Sense Enterprise that can be used by identified or anonymous users. In Qlik Sense, capacity-based licenses are either based on Analyzer Capacity access, or Login access pass allocated with a Token license.

Access types

Access types licenses are the Professional and Analyzer Users licenses (user-based) and Analyzer Capacity licenses (capacity-based). You can combine these for a subscription based license if you use the signed license key when your deployment is activated. You can combine only user-based licenses if you are using a perpetual license.

After changing to a license with a signed key, you cannot return to using the old serial and control number license model.

Professional and Analyzer Users license

A Professional and Analyzer Users license is composed of Professional and Analyzer access types.

- Professional access (user-based) is allocated to an identified user to allow the user to access streams and apps within a Qlik Sense site. The professional access is intended for users who need access to

all features in a Qlik Sense installation. A user with professional access can create, edit, and publish sheets or apps, and make full use of the available features, including administration of a Qlik Sense site.

- Analyzer access is allocated to an identified user to allow the user to access streams and apps in the hub. The analyzer access is intended for users who consume sheets and apps created by others. A user with analyzer access cannot create, edit, or publish sheets or apps, but can create and publish stories, bookmarks and snapshots based on data in apps. The user can also create bookmarks, print objects, stories, and sheets, and export data from an object to Excel.

Analyzer Capacity license

An Analyzer Capacity license is composed of Analyzer Capacity access type.

- Analyzer capacity is a consumption-based license type, which is similar to analyzer access regarding available features. Users can access streams and apps in the hub and consume sheets and apps created by others. Analyzer capacity access allows users to create and publish stories, bookmarks, and snapshots based on data in apps. Creating, editing, or publishing sheets or apps is not possible. With an analyzer capacity license, you subscribe to analyzer time, a defined amount of minutes per month (calendar date). These minutes are shared between users and can be consumed by anyone who is part of the user group, including anonymous users. Consumption is measured in units of 6 minutes. For each new 6-minute period, a unit is consumed.

Dynamic access assignment

Dynamic access assignment is available for Qlik Sense Enterprise on Kubernetes and Qlik Sense Enterprise on Cloud Services deployments and is managed in the management console.

To simplify assignment of access to users, you can enable dynamic assignment. Choose between four options:

- Dynamic assignment enabled for both professional and analyzer access:
  Professional access is assigned, if available, otherwise analyzer access. If neither of those are available, analyzer capacity is assigned, if available.
- Dynamic assignment enabled only for professional access:
  Professional access is assigned, if available, otherwise analyzer capacity is assigned, if available.
- Dynamic assignment enabled only for analyzer access:
  Analyzer access is assigned, if available, otherwise analyzer capacity is assigned, if available.
- Dynamic assignment disabled for both professional and analyzer access:
  Analyzer capacity access is assigned, if available.

You can upgrade from analyzer access to professional access, but not downgrade from professional to analyzer.

If you change to a new license key, all your assignments are removed, because they are associated with the license, not the tenant. However, if you start using the old license key again, the assignments will be present.

Token

You use tokens to allocate access passes to users so that they can access Qlik Sense. The License Enabler File (LEF) determines the number of tokens that you can allocate to different access passes. A user without an access pass cannot access apps.

The token license is available only to customers with existing Qlik Sense Token licenses.

There are two types of access passes that can be allocated using tokens:

- User access pass (user-based) is assigned to unique and identified users allowing them unlimited access to apps, streams, and other resources.
- Login access pass (capacity-based) allocates a block of passes to a group for infrequent or anonymous access. Allows full access for a limited period.

When you allocate tokens, the number of available tokens is reduced. Each access type costs a certain number of tokens, and if the token balance is zero or insufficient, you cannot allocate more to the access types. You can free up tokens and choose to use the tokens differently. The number of tokens for the Qlik Sense site can be increased or decreased by activating a new license.

Core-based site

Qlik Sense Enterprise core-based sites are licensed based on the number of CPU cores on which the software will operate. The license is administered using a License Enabler File (LEF), which limits the maximum number of cores on which the Qlik associative engine and its components may operate. A Core means a single processing unit within a processor or CPU, whether physical or virtual, including a vCPU or virtual core, which is capable of executing a single software thread at a time.

License Enabler File

In Qlik Sense there are two alternative license models: the serial and control number and the signed license key. The License Enabler File (LEF) defines the terms of your license and the access types that you can allocate to users.

When licensing Qlik Sense using a serial and control number, the LEF can be downloaded when the serial number and the control number have been entered in the Qlik Management Console (QMC). The LEF can also be pasted directly into the QMC, if, for example, no network connection is available. There are two license types that can be activated using a serial and control number: Professional and Analyzer Users licenses, and Qlik Token licenses.

When licensing Qlik Sense using a signed key, the LEF file is stored in the License Backend.

If you want to set up Qlik Cloud Services or Qlik Sense Enterprise on Kubernetes, please contact your Qlik representative or Qlik Support to obtain a valid license for the setup.

Professional and Analyzer Users license

Professional and Analyzer Users licenses grant a predefined number Professional and Analyzer (user-based) access type allocations. The LEF file determines the allocation of the access types.

Analyzer Capacity licenses (capacity based) can only be licensed using a signed key. When combining professional, analyzer, and analyzer capacity access types in the same Qlik Sense Enterprise installation, you must license it using a signed key.

Token license

You use tokens to allocate access passes to users so that they can access Qlik Sense. The License Enabler File (LEF) determines the number of tokens that you can allocate, and holds the number of tokens available for the central node in a site. This means that a Qlik Sense site needs at least one (1) LEF. A user without an access pass cannot access apps.

The token license is available only to customers with existing Qlik Sense Token licenses.

You cannot use QlikView CAL-based licenses with Qlik Sense as the tokens are not compatible with the Client Access Licenses (CALs) used in QlikView.

Increase in tokens

When the number of tokens in the LEF increases (for example, when buying additional tokens), the new tokens are added to the pool of unallocated tokens that can be used to allocate access passes that allow users to access Qlik Sense.

Decrease in tokens

When the number of tokens in the LEF decreases, the following happens:

1. Unallocated tokens are removed.
2. If step 1 is not enough to meet the decreased number of tokens in the LEF, any tokens that are freed up by removal of access passes cannot be used for new allocations until the number of allocated tokens is below the new number set in the LEF.

Access assignment

Qlik Sense Enterprise licenses are based either on access types, or on tokens. Depending on your license, you can allocate either access types or access passes to users, to allow them to access Qlik Sense.

- Access types licenses are the Professional and Analyzer Users licenses (user-based) and Analyzer Capacity licenses (capacity-based).
  With a Professional and Analyzer Users license you can allocate professional access and analyzer access.
  With an Analyzer Capacity license you can allocate analyzer capacity access, where consumption is time based (analyzer time).

- With a Qlik Sense Token license you use tokens to allocate access passes to users. You can allocate user access and login access.

Access types

Professional and Analyzer Users licenses and Analyzer Capacity licenses grant a predefined number of access allocations. The License Enabler File (LEF) defines the terms of your license and the access types that you can allocate to users. You can combine these for a subscription based license if you use the signed license key when your deployment is activated. You can combine only user-based licenses if you are using a perpetual license. You must use a license with a signed key if you are licensing analyzer capacity access.

Professional access

Professional access is allocated to an identified user to allow the user to access streams and apps within a Qlik Sense site. The professional access is intended for users who need access to all features in a Qlik Sense installation. A user with professional access can create, edit, and publish sheets or apps, and make full use of the available features, including administration of a Qlik Sense site.

For Qlik Sense installations licensed with a serial and control number, if you remove professional access allocation from a user, the access type is put in quarantine, if it has been used within the last seven days. If it has not been used within the last seven days, the professional access is released immediately. You can reinstate quarantined professional access, to the same user, within seven days.

Quarantine is not applicable for Qlik Sense installations licensed using a signed key.

The maximum number of parallel user connections for a single user of this type of access pass is five (5).If you use a license with a signed license key, accessing the QMC also counts and adds to the maximum number of parallel sessions, which is five. To avoid unnecessary session consumption, the root admin should not be allocated any type of access.

When a user with the maximum number of parallel user connections ends a connection (for example, by logging out) five minutes must pass before the user can use the access pass to add another connection (for example, by logging in).

Analyzer access

Analyzer access is allocated to an identified user to allow the user to access streams and apps in the hub. The analyzer access is intended for users who consume sheets and apps created by others. A user with analyzer access cannot create, edit, or publish sheets or apps, but can create and publish stories, bookmarks and snapshots based on data in apps. The user can also create bookmarks, print objects, stories, and sheets, and export data from an object to Excel.

For Qlik Sense installations licensed with a serial and control number, if you remove analyzer access allocation from a user, the access type is put in quarantine, if it has been used within the last seven days. If it has not been used within the last seven days, the analyzer access is released immediately. You can reinstate quarantined analyzer access, to the same user, within seven days.

Quarantine is not applicable for Qlik Sense installations licensed using a signed key.

The maximum number of parallel user connections for a single user of this type of access pass is five (5).When a user with the maximum number of parallel user connections ends a connection (for example, by logging out) five minutes must pass before the user can use the access pass to add another connection (for example, by logging in).

Analyzer capacity access

Analyzer capacity is a consumption-based license type, which is similar to analyzer access regarding available features. Users can access streams and apps in the hub and consume sheets and apps created by others. Analyzer capacity access allows users to create and publish stories, bookmarks, and snapshots based on data in apps. Creating, editing, or publishing sheets or apps is not possible.

With an analyzer capacity license, you subscribe to analyzer time, a defined amount of minutes per month (calendar date). These minutes are shared between users and can be consumed by anyone who is part of the user group, including anonymous users. Consumption is measured in units of 6 minutes. For each new 6-minute period, a unit is consumed.

Access passes

With a Qlik Sense Token license you use tokens to allocate access passes to users. The License Enabler File (LEF) determines the number of tokens that you can allocate to different access passes. A user without an access pass cannot access apps.

User access pass

This type of access pass allows a unique and identified user to access the hub.

The access pass is valid within an entire Qlik Sense site. For example, if a user first connects to a node in the USA and then, at a later stage, connects to a node in the UK, the user consumes the same access pass, if the two nodes are connected to the same central node.

The maximum number of parallel user connections for a single user of this type of access pass is five (5).When a user with the maximum number of parallel user connections ends a connection (for example, by logging out) five minutes must pass before the user can use the access pass to add another connection (for example, by logging in).

One (1) token corresponds to one (1) access pass. The access passes are allocated using the Qlik Management Console (QMC).

You can have both a user access pass and the possibility to consume login access passes. If you have five active sessions, opening an additional session will consume from your login access passes.

Removing user access pass allocation

When a user access pass is removed, it enters a quarantine for seven (7) days, counting from the last time that the access pass was used. For example, if the access pass is used on January 10, the tokens used to allocate the access pass are not available for new allocations until January 18. During the quarantine period, the original allocation of the access pass can be reinstated, which means that the quarantine period ends and the user can start using the access pass again.

Login access pass

This type of access pass allows an identified or anonymous user to access the hub for a maximum of 60 continuous minutes per 28-day period. If the user exceeds the 60 minutes time limitation, the user connection does not time out. Instead, another login access pass is used. If no more login access passes are available, the user connection is discontinued.

- If an identified user is disconnected, the user can re-connect and continue to use the same access pass, if re-connecting within the 60 minutes.
- If an anonymous user is disconnected, the user gets a new access pass when re-connecting.

The login access pass tracks the number of logins and runs over 28 days. For example, if 1000 logins are assigned to Group A, the users in Group A can use 1000 logins over 28 days. If 100 logins are consumed on Day 1, the 100 logins are available again on Day 29.

The maximum number of parallel user connections for a single user of this type of access pass is five (5). Note that this only applies to identified users. An anonymous user can only have one (1) user connection. When a user with the maximum number of parallel user connections ends a connection (for example, by logging out) five minutes must pass before the user can use the access pass to add another connection (for example, by logging in). However, a user can have more connections than allowed by a single access pass by consuming additional access passes.

One (1) token corresponds to ten (10) access passes. The access passes are allocated using login access groups in the QMC.

App reloads will extend the session and consume access passes also when the app is not actively used. If a browser page is open with an app, app reloads will result in additional access pass consumption.

Removing login access pass allocation

When a login access group is removed, the tokens used to allocate the access pass become available in accordance to the following procedure:

1. For every ten (10) **unused** login access passes, one (1) token is freed up.
2. For every ten (10) login access passes that leave the **used** state after the period specified in the *Access assignment (page 11)* section above has passed, one (1) token is freed up.

1.2 Before you install Qlik Sense Enterprise on Kubernetes

To successfully plan and prepare for your Qlik Sense Enterprise on Kubernetes deployment, do the following:

System requirements

Check that your environment fulfills the system requirements.

Supported browsers
: Check that your browsers are supported.

Multi-cloud services
: Understand the Qlik Sense services.

System requirements for Qlik Sense Enterprise on Kubernetes

This section lists the requirements that must be fulfilled by the target system in order to successfully install and run Qlik Sense Enterprise on Kubernetes.

Kubernetes environments	The Kubernetes environment must have Internet access to the Qlik Helm and Container Image repository. Kubernetes service vendors: • Microsoft Azure using Azure Kubernetes Service (AKS) • Amazon Web Services (AWS) using Amazon Elastic Container Service for Kubernetes (EKS) • Amazon Web Services (AWS) deployed via Kubernetes Operations (KOPs) • Google Cloud using Google Kubernetes Engine (GKE) • Red Hat OpenShift 4+ Non-managed Kubernetes deployments: • Kubernetes cluster greater than v1.10.x and less than v1.16.x
Kubernetes package manager	Helm greater than v2.12.0 and less than v2.15.x
Local/Evaluation/Test environment	Windows: Minikube v0.33 + Red Hat MiniShift v1.21.0+ Mac: Docker for Desktop with Kubernetes enabled: v2.0.0.3
Database	MongoDB 3.6+
File system	Storage attached to the cluster that supports ReadWriteMany. This can be configured as a Storage Class or a Persistent Volume Claim
Processors (CPUs)	Minimum 4 cores (additional depending on data volumes)
Memory	Minimum 8 GB (additional depending on data volumes)
Disk space	5 GB total required to install
IDP	For user authentication an OIDC compatible IDP is required

Supported browsers in Qlik Sense Enterprise on Kubernetes

Qlik Sense is designed to work on the platform and web browser combinations described in this section, using default browser settings.

Each Qlik Sense release is tested for compatibility with the latest publicly available browser versions. Due to the frequency of browser version updates, Qlik does not include specific browser version numbers in the system requirements.

Each Qlik Sense release is compatible with and supported on the latest iOS versions that are publicly available at the time of the Qlik Sense release. Due to the frequency of iOS version updates, Qlik does not include specific iOS version numbers in the system requirements.

Microsoft Windows 7

- Microsoft Internet Explorer 11
- Google Chrome
- Mozilla Firefox (requires hardware acceleration, not supported in virtual environments)

Microsoft Windows 8.1

- Microsoft Internet Explorer 11
- Google Chrome
- Mozilla Firefox (requires hardware acceleration, not supported in virtual environments)

Microsoft Windows 10

- Microsoft Edge
- Microsoft Internet Explorer 11
- Google Chrome
- Mozilla Firefox (requires hardware acceleration, not supported in virtual environments)

Apple Mac OS X 10.11 and 10.12

- Apple Safari 10 or later
- Google Chrome
- Mozilla Firefox (requires hardware acceleration, not supported in virtual environments).

Microsoft Windows Server 2012 R2

- Microsoft Internet Explorer 11
- Google Chrome
- Mozilla Firefox (requires hardware acceleration, not supported in virtual environments)

Microsoft Windows Server 2016

- Microsoft Internet Explorer 11
- Google Chrome
- Mozilla Firefox (requires hardware acceleration, not supported in virtual environments)

CefSharp embedded browser v55 or later (CefSharp allows you to embed the Chromium open source browser inside .Net apps)

Microsoft Windows Server 2019

- Microsoft Internet Explorer 11
- Google Chrome
- Mozilla Firefox (requires hardware acceleration, not supported in virtual environments)

CefSharp embedded browser v55 or later (CefSharp allows you to embed the Chromium open source browser inside .Net apps)

iOS

Version 11.2 or later (script editing is not supported on tablet devices).

Qlik Sense version: Qlik Sense EnterpriseSeptember 2017 or later.

Supported devices:

- iPad Air or later
- iPhone 5S or later

Supported browsers:

- Apple Safari
- Google Chrome
- VMware browser (using AirWatch per-app VPN)
- BlackBerry Access 2.9.1 or later (using BlackBerry Dynamics platform)

iOS 11.3 is required for using BlackBerry Access browser.

Android

Version 6.0, 7.1, 8.1 and 9.0 (script editing is not supported on tablet devices):

- Google Chrome
- BlackBerry Access 2.9.1 or later (using BlackBerry Dynamics platform)

Windows 10 phone

- Microsoft Edge

Minimum screen resolution for desktops and laptops is 1024x768; tablets is 1024x768; small screens is 320x568.

Multi-cloud services

You have several options when deploying a Qlik Sense Enterprise on Windows environment. The services that you need to run in a multi-cloud deployment can be categorized as follows.

Typically the services running in a QCS deployment are similar to those running in a Qlik Sense Enterprise on Kubernetes deployment but are not accessible, because Qlik manages the infrastructure. You can connect to QCS SaaS but do not have the same configuration options as a Kubernetes deployment.

Services on Windows deployments

The services listed below are required if you use the multi-cloud capabilities in a Qlik Sense Enterprise on Windows deployment.

Service	Description
App Distribution Service	Distributes apps and associated metadata to defined distribution targets, based on policy-based app distribution rules.
Hybrid Deployment Service	Stores configuration details including credentials and URLs for all target environments in a multi-cloud deployment.
Hybrid Setup Console Service	Multi-cloud Setup Console UI functions for managing target environments configured in a multi-cloud deployment including credentials and service URLs.
Resource Distribution Service	Publishes installed extensions and themes to the Resource Library in each cloud environment.

Services on Kubernetes and Qlik Cloud Services deployments

The services that you run in Qlik Sense Enterprise on Kubernetes can vary depending on your deployment requirements.

Service	Description
Chronos	Scheduler back end service.
Cloud hub	Serves the hub functionality to users in Qlik Cloud Services and Qlik Sense Enterprise on Kubernetes.
Collections	Organizes and structures content supplied to the hub. It also applies access control rules.
datafiles	Allows user to upload/manage data files that can be accessed during reload of an app.
edge-auth	Service that works together with external Identity Providers to authenticate users upon entry to the deployment. Also manages tickets that authorize secure access to internal resources.
elastic-infra	A collection containing non-Qlik services: MongoDB, Redis, and nginx-ingress. It bootstraps an elastic-infra deployment on a Kubernetes cluster using the Helm package manager. It starts up the basic resources needed to connect all the components and functionality required in a cloud environment.
Engine	Handles all application calculations and logic.

Feature Flags	Responsible for toggling features on and off in advanced scenarios.
Insights	REST service for the functionality behind the “Share” dialog on a sheet. The service is responsible for sharing Qlik Sense insights by generating, tracking and serving persistent permalinks to the shared resources. Permalinks can be shared in various social media.
Licenses	The license service is used to enforce user licensing in Qlik Cloud Services and Qlik Sense Enterprise on Kubernetes.
Locale	Handles user locale selection for the client.
Mira	Provides a discovery service for Engines in the deployment, their current health, and availability of applications.
ODAG	Service for on-demand app generation of Qlik Sense apps.
Policy Decision	Processes a set of rules on Qlik Cloud Services and Qlik Sense Enterprise on Kubernetes that perform ABAC security evaluation against Qlik objects (for example, apps). It is sometimes referred to as the Rules Service. It uses a REST API for the rules engine and management API for rule based policies and replaces the QRS Rules Engine, Policy Decision Service.
qix-data-connection	Handles engine requests for connection management.
qix-sessions	Responsible for routing user session traffic to the Engine services.
Reporting	Implements the productions of Reports with data and chart images.
Resource Library	A general-purpose resource storage service for supporting content such as themes and extensions.
Tenant	Used to store and return tenant (user) information.
temporary-contents	Manages resources that are made available temporarily.
User	Responsible for managing and retrieving user information.
Sense Client	The Desktop and web browser instance of the Qlik Sense client run by developers on Qlik Sense Enterprise and by consumers on QCS and Qlik Sense Enterprise on Kubernetes.

1.3 Qlik Sense Enterprise on Kubernetes deployments

This diagram shows an example of a Qlik Sense Enterprise on Kubernetes deployment with a single Kubernetes cluster connected to a Qlik Sense Enterprise on Windows node. The cluster contains one or more of the Qlik Sense microservices such as the Engine or other services deployed across a set of nodes. This deployment provides the ability to scale up the number of apps (read only) for user consumption. The Kubernetes cluster, which is deployed within a public or private cloud, shares data volumes and a MongoDB instance. An Identity Provider (IdP) authenticates users while QSE authorizes access to multi-

cloud apps using built-in security rules. The IdP allows the same named users to access content in Qlik Sense Enterprise and the cloud environment, subject to security rules. The Kubernetes cluster, public or private cloud, and network infrastructure are all managed by the customer.

CSRF security for Qlik Sense Enterprise on Kubernetes

Cross-site request forgery (CSRF) is when someone attacks a user's web application by taking advantage of that user's authentication. For example, if a user is already authenticated on a secure web application and they click a malicious link during their web session, an attacker can then use their authentication to perform tasks or actions without the user's permission or knowledge.

To ensure that Qlik Sense Enterprise on Kubernetes APIs are protected against CSRF security risks, Qlik has implemented token-based anti-CSRF security for its APIs that will prevent CSRF attacks.

This token is generated on the server-side and is linked to a specific session by the web server, which is then used as a hidden value in every web application form. Since the token is on the server side and not in the web session, a hacker has no way to get that token because they do not have access.

2 Preparing for Qlik Sense Enterprise on Kubernetes

Qlik Sense Enterprise on Kubernetes is deployed to a Kubernetes cluster in the form of a set of container images in a package provided as a Helm chart. To be able to install, the following items are required to be in place as a minimum (also refer to the system requirements for more details).

- A running Kubernetes cluster - this can be run locally for development purposes or deployed to cloud vendors including AWS, Google Cloud and Azure.
- The Kubernetes cluster must have access to file storage to persist data. This should be provided as a Storage Class that allows **`readwritemany`** access. You will need the name of this Storage Class when installing.
- You also require a license key for Qlik Sense Enterprise and this must be in the signed license key format. A serial number and control number cannot be used.
  Contact Qlik support if you do not have this version of your license.

The Qlik documentation does not cover the installation and configuration of a Kubernetes cluster and you should review the documentation for this at either https://Kubernetes.io or for the cloud vendor or product you are using.

You will also need the following tools installed on your local machine to interact with your Kubernetes environment, issue commands and deploy software:

- Kubectl - Install kubectl on the machine you will run admin commands from. You can find further details for your operating system at https://Kubernetes.io.

  This can point to different clusters if you have more than one. Ensure that the commands go to the right Kubernetes instance.

- Helm - Helm is a package manager built for Kubernetes. It has a concept known as charts, used to define what services are required, what images are used, and default settings when the run in the Kubernetes cluster. It is used to push the Qlik Sense package into Kubernetes and relies on kubectl, so it must be installed on the same machine as kubectl. Qlik uses helm to define a default chart to make deployments simple for customers.
  To install Helm on your local machine follow the instructions for your operating system at https://docs.helm.sh/.

2.1 Preparing your local tools

Once you have set up your Kubernetes cluster, you must prepare your local tools to work with your Kubernetes cluster. To prepare your local tooling you must:

- Bind kubectl to your Kubernetes cluster
- Add Qlik's helm chart repository

- Initialize helm to work with your Kubernetes cluster

Before you begin you should have the following installed on your local machine:

- Kubectl
- Helm

To bind kubectl to your Kubernetes cluster:

1. Verify that kubectl is pointing to your Kubernetes cluster using the following command:
   ```
   kubectl config current-context
   ```
2. If kubectl is not pointing to your Kubernetes cluster, use the following command to get a list of available clusters:
   ```
   kubectl config get-clusters
   ```
3. Set the kubectl to point to the desired cluster using the following command:
   ```
   kubectl config set-cluster <cluster-name>
   ```

To add Qlik's helm chart repository:

1. Run the following command to add Qlik's helm chart repository to Helm. This is where Qlik Sense is pulled from:
   ```
   helm repo add qlik https://qlik.bintray.com/stable
   ```
2. Use the following command to get a list of all configured repositories and verify that the Qlik helm chart repository was successfully added:
   ```
   helm repo list
   ```

To use helm to with your Kubernetes cluster it needs to be initialized to create the helm Tiller pod that handles installations:

1. The following command is used to do this in simple cases
   To use helm to deploy into Kubernetes, the helm Tiller pod is added to the Kubernetes cluster first. The following command is used to do this in simple cases:
   ```
   helm init --wait
   ```
2. If the Kubernetes cluster has security features such as RBAC enabled then the following commands should be run in addition:
   ```
   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 Installing Qlik Sense Enterprise on Kubernetes

Once you have set up your Kubernetes cluster and prepared your local tools, you are ready to install Qlik Sense Enterprise into your Kubernetes cluster.

To recap, as a minimum before you install you will have:

- Set up a Kubernetes cluster and added a Storage Class with **`readwritemany`** storage
- Prepare your local tools to work with your Kubernetes cluster

The steps below cover the steps to install a first simple install. This includes deploying a test instance of MongoDB and a simple IDP to get you running. To move it to a production ready state, you should review the additional topics for the following areas:

- Understanding and configuring an IDP to authenticate users
- Configuring a separate MongoDB instance
- Viewing and handling logs

3.1 Providing configuration settings

When installing Qlik Sense Enterprise on Kubernetes you can specify settings to the installer in two ways:

- As parameters in the `helm install` command.
- Referencing the settings in a **values.yaml** and using this in the `helm install` command.

Storing the configuration settings in a **values.yaml** allows you to reuse the settings in multiple deployments and add new config sections simply. This can also be version controlled.

Use of a **values.yaml** file will be used predominately in the Qlik help. You can find more information about YAML files online on sites such as https://en.wikipedia.org/wiki/YAML.

3.2 Installing Qlik Sense

Complete the following steps to install Qlik Sense:

1. Create a text file called **values.yaml**.
    a. Add the following content to the file:
    ```
    #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"
    ```

> *If devMode.enabled is set to true, a MongoDB instance is deployed inside of your Qlik Sense Enterprise on Kubernetes in Kubernetes for development and testing purposes only.*

b. Add the following content to point the services requiring storage to the Kubernetes Storage Class, update the name of the Storage Class as needed.

> *If you are using Kubernetes via Docker for Desktop or Minikube then you should not add this section.*

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

c. If you are using Minikube for test purposes, you should review and add the additional configuration here: *Using Minikube (page 25)*.

d. Save the file.

2. Qlik Sense will dynamically create engines to run scheduled reloads. To be able to do this, the engine needs to be configured in Kubernetes as a custom resource.
   This step only needs to be done once for a cluster. Run the following command to install custom resource definitions used by dynamic engines:
   ```
   helm install --name qliksense-init qlik/qliksense-init
   ```
3. The next step is to install the Qlik Sense package. Run the following command:
   ```
   helm install -n qliksense qlik/qliksense -f values.yaml
   ```
   The software now starts deploying to the Kubernetes cluster, including downloading all the images and running them.
4. You can now use kubectl to check the progress. Run the following command:
   ```
   kubectl get pods
   ```
   If your deployment was successful you will see something similar to this:

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)				

> *It typically takes a few minutes to initialize and show a status of "Running".*

> *If services do not start check the log files of the service for more information. If some services remain in a pending state, check that the Kubernetes cluster has* ***`readwritemany`*** *storage available as a storage class and that is it correctly referenced in the YAML.*

3.3 Accessing the deployment

To connect to the hub and confirm the install you need to obtain the URL for the install inside the Kubernetes cluster. This can vary depending on the configuration and / or vendor as follows.

For most cloud vendors (for example AWS, GCP or Azure) the IP address will be generated during the installation automatically. You can find the address by running the following command:

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

For Docker Desktop, the IP will typically be the machine loop-back address `127.0.0.1`. For Minikube, the IP can be obtained by running the command `minikube ip`.

Creating an alias to the IP address

In this simple deployment an example Identity Provider is automatically configured. This allows you to login to the hub with some sample accounts. This service will by default only listen to the URL *https://elastic.example*, it cannot be browsed to on the IP address alone. To this end you must add a Host file entry to point the IP address from above to the alias on *elastic.example*.

This step is only required when running a basic example, a production example will use a real Identity Provider during which the correct DNS entry for the cluster will be used.

The simple IdP is for test purposes only and you should configure a full IdP by reviewing Setting up identity providers (page 29).

Logging in and applying the license

You can now browse to the hub at *https://elastic.example*. You will be asked to login and you can use the sample account of **harley@qlik.example** with a password of **Password1!** .

You should now navigate to *https://elastic.example/console* to apply the license before being able to create applications.

3.4 Using Minikube

Minikube is a test environment that allows for a Kubernetes environment to run locally on a Windows, Mac or Linux machine. It can be used to run Qlik Sense Enterprise on Kubernetes for local test purposes.

Minikube does not support the LoadBalancer resource that all other Kubernetes providers use and requires running on a different port, an additional configuration section is required when using it.

These additional configuration items should not be used with any other Kubernetes provider as they will not work.

Preparing Minikube

Before you start, review the installation documentation for Minikube for your operating system here: https://kubernetes.io/docs/setup/minikube/.

When you start Minikube, ensure that it has enough resources by running the following (you may need to run `minikube delete` first):

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

Additional configuration when using Minikube

With Minikube running, follow the installation steps in *Installing Qlik Sense Enterprise on Kubernetes (page 23)* but add the following additional section into **values.yaml**.

```
# 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
```

Accessing the installation on Minikube

Once it is started, you can get the IP address the Kubernetes cluster is running on with the following command:

```
minikube ip
```

When using Minikube you must specify the port 32443 in the URLs used to access the product IP, for example https://elastic.example:32443.

3.5 Installing Qlik Sense Enterprise on Kubernetes to a Red Hat OpenShift platform

To install Qlik Sense Enterprise on a Red Hat OpenShift Kubernetes platform there are a few additional considerations:

- OpenShift default namespace is **myproject**
- OpenShift typically comes with its own docker image registry. All charts now support global.imageRegistry so that customers can point all image pulls from their own image registry.
- OpenShift does not allow containers to run as root by default.
- OpenShift discourages cluster wide access.
- OpenShift expects services to be exposed by routes.
- OpenShift has it's own certified container registry and discourages non-certified containers.
- OpenShift deployments do not typically require internet access. Qlik Sense Enterprise on Kubernetes does require internet access.

Once you have set up your Kubernetes cluster and prepared your local tools, you are ready to install Qlik Sense Enterprise into your Kubernetes cluster. See *Installing Qlik Sense Enterprise on Kubernetes (page 23)*.

Qlik supports deployments of Qlik Sense Enterprise on Kubernetes to Red Hat OpenShift platforms, however the deployment is not currently certified by Red Hat.

Using MiniShift

To get an OpenShift platform running use MiniShift to create a virtual machine (VM) and provision a local, single-node OpenShift cluster in the VM. RedHat provides MiniShift as a simple to run OpenShift platform. MiniShift is available at https://github.com/minishift/minishift.

1. Download the latest version of MiniShift.
2. Unzip it to a directory, and add this directory to your path.
3. Install helm (mandatory) and kubectl (optional).
4. Set the resource settings for MiniShift to ensure that there are adequate resources to fully deploy Qlik Sense Enterprise on Kubernetes, using the following commands:
   ```
   minishift config set memory 8192
   minishift config set cpus 4
   ```
5. Start up MiniShift using the following command:
   ```
   minishift start --vm-driver=virtualbox
   ```
6. Configure your environment using the following command:
   ```
   minishift oc-env
   ```
7. Create a new project. A project is the OpenShift equivalent of a kubernetes namespace.

Note that in this example we use `oc` instead of `kubectl`. In most cases they do the same thing, however `oc` does some things that cannot be done with `kubectl`.

```
oc new-project tiller
```

8. Set a variable so helm knows where to find tiller.
```
$Env:TILLER_NAMESPACE="tiller"
```
9. Configure the helm client.
```
helm init --client-only
```
10. Installing tiller on OpenShift requires a special process. OpenShift provides a yaml file to do that. Run the following command:
```
oc process -f https://github.com/openshift/origin/raw/master/examples/helm/tiller-template.yaml -p TILLER_NAMESPACE="$Env:TILLER_NAMESPACE" -p HELM_VERSION=v2.9.1 | oc create -f -
```
11. To confirm it is ready run the following command:
```
oc rollout status deployment tiller
```
12. Create a project for the Qlik deployment using the following command:
```
oc new-project qlik
```
13. To configure security you must log in as a system admin using the following command:
```
oc login -u system:admin
```
14. Configure the security settings needed by running the following commands:
```
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. Define the storage and make it writable using the following command:
```
minishift ssh -- "sudo chmod -R o+rwx /var/lib/minishift/base/openshift.local.pv*"
```
16. Log in as a developer using the following command:
```
oc login -u developer
```
17. Add the Qlik helm repository and update it using the following commands:
```
helm repo add qlik https://qlik.bintray.com/stable
helm repo update
```
18. You can now deploy Qlik Sense Enterprise on Kubernetes using the minishift.yaml file.
```
helm upgrade --install qsefe qlik/qsefe --set devMode.enabled=true,engine.acceptEULA="yes" -f C:\dev\minishift.yaml
```
19. To monitor deployments you can run `kubectl get pods`, or `oc get pods`.

4 Setting up Qlik Sense Enterprise on Kubernetes after installation

This section guides you through the process of setting up your Qlik Sense Enterprise on Kubernetes site after installing. You can configure the server to fit with your organization's particular needs. Below are the common task most deployments will require.

An implementation of Qlik Sense Enterprise on Kubernetes can vary depending on the configuration required. The following pages detail on the key elements required to do a production implementation:

- *Distributing apps to Qlik Sense Enterprise on Kubernetes (page 29)*
- *Setting up identity providers (page 29)*
- *Configuring certificates in your Qlik Sense Enterprise on Kubernetes deployment (page 43)*
- *Configuring MongoDB in Qlik Sense Enterprise on Kubernetes (page 45)*

When you install Qlik Sense Enterprise on Kubernetes, we provide a default set of service keys for service to service communication. We recommend that you rotate these default keys as part of securing your install for production.

4.1 Distributing apps to Qlik Sense Enterprise on Kubernetes

Once you have Qlik Sense Enterprise on Kubernetes running, you can distribute apps into it from your Qlik Sense Enterprise on Windows deployment. To distribute apps, complete the following steps:

1. Create a Deployment in the Multi-cloud Setup Console on your Qlik Sense Enterprise on Windows deployment.
   See *Multi-Cloud Setup Console - start page* and *MSC - Deployments*.
2. Create a distribution policy to decide which applications should be distributed.
   See *Distribution policies - introduction*.
3. Publish the application setting the **collection**, **userswithaccess** or **groupswithaccess** properties on the app.
   See *Publishing apps to cloud hub collections*.

Once you have distributed the apps, you will be able to open them from the hub.

4.2 Setting up identity providers

An identity provider (IdP) manages identity information for users and provides authentication services. The identity provider enables single sign-on (SSO) so that you can access other websites, without having to log in

repeatedly. In contrast to on-premise technologies, such as Active Directory and LDAP, identity providers also offer a consistent and governed experience when accessing cloud services, eliminating the need to create accounts for each new service.

> *If user accounts are stored in Active Directory, the IdP can still enable integration into cloud software.*

In Qlik Sense Enterprise on Cloud Services, Qlik Sense Enterprise on Kubernetes or in a multi-cloud deployment, an IdP delivers the following:

- Secure authentication of a user and a common identity (user ID and groups) passed between all deployments.
- Common user identity to assign a license to (to avoid double use).
- Common user ID and attributes, such as groups, to use when applying access control to content.

Example: IdPs in a multi-cloud deployment

IdP requirements

Both Qlik Cloud Services and Qlik Sense Enterprise on Kubernetes integrate with an IdP using the OpenID Connect (OIDC) standard. This is a standard that allows both interactive login, where a user logs in via a browser, and automated login, using APIs via a software product.

Qlik Sense Enterprise on Windows currently does not support OIDC, but supports SAML, or any method that allows a consistent user identity to the one provided by the IdP.

> *In summary, an IdP for multi-cloud must support both OIDC and SAML.*

The following is required from the IdP to be able to set up Qlik Sense Enterprise on Kubernetes to use it:

- **discoveryUrl**: the OpenID Connect Discovery URL which allows applications, such as Qlik Sense, to use the IdP with minimal configuration.
- **clientId**: uniquely identifies the client from the IdP.
- **clientSecret**: the secret that the client uses along with the client ID to authentication with the IdP.
- **realm**: the name to associate with the IdP.
- **hostname**: the hostname that is used for the deployment of Qlik Sense Enterprise on Kubernetes.

These values are added to the **values.yaml** file under the identity-providers section when installing Qlik Sense Enterprise on Kubernetes.

Step-by-step examples of this configuration are provided for the following IdP vendors:

- Okta
  See *Setting up Okta* .

- Auth0
  See *Setting up Auth0*.
- ADFS
  See *Setting up ADFS*.

Setting up ADFS

ADFS is an authentication and authorization platform.

You can configure ADFS as an identity provider (IdP) for use with Qlik Sense Enterprise on Kubernetes (QSEoK) and Qlik Sense Enterprise on Windows (QSEfW). You will create an application group, a server application, and a Web API to be used for interactive login (QSEoK). You will also map claims from Active Directory to the ID token.

Creating required ADFS resources for QSEoK for interactive logins

For setting up ADFS, you need an application group and a server application.

> *The following procedures are examples using ADFS 10. Please review the ADFS documentation for more information and latest instructions.*

Adding an application group and creating a server application

Do the following:

1. Open the **Add Application Group Wizard**.
2. Enter a name for the application group.
3. For **Template**, select **Server application**.
4. Click **Next**.
   The **Server application** page is opened.
5. Enter a name for the application.
   Example: *1234567890*
6. Enter a client identifier for the application, and note it down, it will be used as client ID.
   Example: *https://adfs.elastic.example/1234567890*.

   > *In this example, https://adfs.elastic.example is the tenant domain and 1234567890 is a unique identifier for the application. The client identifier must be a URL. ADFS will only include custom claims in the id_token for applications with URL IDs, see* Customize claims to be emitted in id_token when using OpenID Connect or OAuth with AD FS 2016.
7. For **Redirect URI**, set the redirect URL to the login callback for the tenant in the format *https://<host>/login/callback/*.
   Example: *https://adfs.elastic.example/login/callback*
8. Optionally, enter a description.

9. Click **Next**.
   The **Configure Application Credentials** page is opened.
10. Select **Generate a shared secret**. Note down this secret, you will not have access to it again. You will use it as client secret.
11. Finish the wizard.

Adding a web API to the application group

You will add a web API to the application group that you created.

Do the following:

1. Open the application group you created earlier.
2. Select **Add application** > **Web API**.
3. Add the client ID from the application group as in identifier.
4. Click **Next**.
   The **Choose Access Control Policy** page is opened.
5. Apply a policy and click **Next**.
   The **Configure Application Permissions** page is opened.
6. For **Permitted scopes**, select the following: *allatclaims*, *email*, *openid*, and *profile*.
7. Finish the wizard.

Configure claims for the id_token

Do the following:

1. Open the application group to edit the web API you created. Open the **Issuance Transform Rules** tab.
2. Create a rule from the rule template **Send LDAP Attributes as Claims**.
3. Select **Active Directory** as the attribute store.
4. Add claims mappings. You may need to type the outgoing claim.
5. Map *Token-Groups - Unqualified Names* to *groups*.
6. Map *Display-Name* to *display_name*.
7. Finish the claims mapping.

Using ADFS as an IdP for Qlik Sense Enterprise on Kubernetes

You can use ADFS as an identity provider for logging into a Qlik Sense Enterprise on Kubernetes tenant using a user from ADFS.

Connecting Qlik Sense Enterprise on Kubernetes with ADFS

Before you start, make sure you have the following:

- ADFS installation
- the required resources configured in ADFS

- Configuration settings from your ADFS application: *discoveryUrl*, *clientId*, and *clientSecret*
- The following values from your hybrid deployer: public key, key ID, and issuer.

> *Many of the code examples contain placeholder values that need to be replaced by your own values.*

You provide configuration to Qlik Sense Enterprise on Kubernetes by using a *values.yml* file. The *values.yml* file should look like the following example:

```
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-----
```

> *It is important to note that the userClaimsFromIdToken flag is set to true. The flag instructs edge-auth to use the claims from the ID token instead of querying for userinfo. This is because ADFS returns very little in the userinfo response and instead includes most information in the ID token.*

You will have to insert your own values for *discoveryUrl*, *clientId*, *clientSecret*, *realm* and *hostname*.

Applying the configuration to your cluster

Use Helm (see https://helm.sh/) to apply the configuration in your *values.yml* file to your Kubernetes cluster:

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

To make sure that your configuration has been applied, you can run the `get values` command to see the resolved configuration:

```
$ 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-----
```

Configure your hosts file

This section is only relevant if there is no DNS.

In order for `<hostname>` to resolve, add the following to your `/etc/hosts` file:

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

Log in to your tenant

You are now set to log into your tenant with a user from your ADFS deployment. In your browser, go to *https://<tenant address>* and you should be redirected to an ADFS login page. After a successful login you reach a home page to which apps are distributed.

Setting up Auth0

Auth0 is an authentication and authorization platform.

You can configure Auth0 as an identity provider (IdP) for use with Qlik Sense Enterprise on Kubernetes (QSEoK) and Qlik Sense Enterprise on Windows (QSEfW).

Creating an Auth0 application and connection for QCS or QSEoK for interactive logins

Create an Auth0 application, and connect it to an Auth0 database connection.

An Auth0 application allows an application, (QSEfW/QCS/QSEoK), to use Auth0 for authentication. An Auth0 connection is a source of users, in this example, a database that you populate with users.

We assume that you have an Auth0 account and tenant created.

The following procedures are examples. Please review the Auth0 documentation for more information and latest instructions.

Creating a new application in Auth0

Do the following:

1. In the left menu in Auth0, open **Applications**.
2. Click **Create application**.
3. Give the application a name, select **Single Page Web Applications** and click **Create**.
4. Optionally, select your web app technology.
5. Select **Settings**.
6. In the box **Allowed Callback URLs**, add the URL to your host in the format *https://<host>/login/callback/*.
7. Scroll down and click **Save changes**.
8. Note down the **Client ID** value.
9. Note down the **Client Secret** value.
10. Scroll to the bottom and select **Advanced Settings**.
11. Select the **Endpoints** tab.
12. Note down the **OpenID configuration** URL for later.

Creating a database connection in Auth0

You will now create a database connection and configure your application to use this connection.

Do the following:

1. In the left menu, select **Connections** > **Database**.
2. Fill in a name for the database connection and click **Create**.
3. In the left menu, select **Applications**.
4. Open the tab **Connections**.
5. Enable the new database connection for your application.

Creating a new user (optional)

Do the following:

1. In the left menu, select **Users**.
2. Click **Create your first user**.
3. Fill in the fields and select the newly created connection.

Creating an Auth0 API and application for programmatic access

Begin by creating the API.

You set up programmatic access so that you can distribute content into Qlik Cloud Services (QCS) or QSEoK.

In Auth0, you will create a new API. In this case, the Auth0 API represents the protected QSEoK resource API. In OAuth terms, you configure Auth0 for the Client Credentials Grant flow.

Begin by creating a new API for your application.

Do the following:

1. In the left menu, select **APIs**.
2. Click **Create API**.
3. Enter an API name.
4. For **Identifier**, enter `qlik.api`.
5. Click **Create**.
6. Go to the **Scopes** tab.
7. Add a new scope with the value `any` in the name and description and click **Add**.

Just like you created an Auth0 application for interactive logins above, you will now create an Auth0 application for programmatic authentication.

Do the following:

1. In the left menu, select **Applications**.
2. Click **Create Application**.
3. Select **Machine to Machine Applications**.
4. Click **Create**.
5. Select the API created above.

6. In the **Scopes** box, select **any**.
7. Click **Authorize**.
8. Select the **Settings** tab. In the **Allowed Web Origins** box, add the URL to your deployment.
9. Note down the **Client ID** value.
10. Note down the **Client secret** value.
11. Scroll to the bottom and select **Advanced Settings**.
12. Click the **Endpoints** tab.
13. Note down the **OAuth Token URL** value.
    This value together with client ID and client secret will be used in the configuration of QSE for Windows when adding a deployment.
14. In the left menu, select **APIs** and open your new API. Select the **Machine to Machine Applications** tab.
15. Verify that your new application has access to your new Auth0 API.

Using Auth0 as an IdP for Qlik Sense Enterprise on Kubernetes

You can use Auth0 as an identity provider for logging into a Qlik Sense Enterprise on Kubernetes (QSEoK) tenant and also for interacting with the tenant programmatically.

Connecting QSEoK with Auth0

Before you start, make sure you have the following:

- Auth0 account
- Auth0 tenant
- Auth0 app, configured with interactive login and programmatic access
- Configuration settings from your Auth0 application: *discoveryUrl*, *clientId*, and *clientSecret*

Many of the code examples contain placeholder values that need to be replaced by your own values.

You provide configuration to QSEoK by using a *values.yml* file. The *values.yml* file should look like the following example:

```
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>" ]
```

You need to enter the values for *discoveryUrl*, *clientId*, *clientSecret*, *realm*, *hostname*, and *id* (claims mapping).

Applying the configuration to your cluster

Use Helm (see https://helm.sh/) to apply the configuration in your *values.yml* file to your Kubernetes cluster:

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

To make sure that your configuration has been applied, you can run the `get values` command to see the resolved configuration:

```
$ 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>"
```

Configure your hosts file

This section is only relevant if there is no DNS.

In order for `<hostname>` to resolve, add the following to your `/etc/hosts` file:

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

Log in to your tenant

You are now set to log into your tenant. In your browser, go to *https://<tenant address>* and you should be redirected to an Auth0 login page. After a successful login you reach a home page to which apps are distributed.

Setting up Okta

Okta is an authentication and authorization platform.

This topic presents how to set up Okta to be used with Qlik Sense Enterprise on Kubernetes (QSEoK) and Qlik Sense Enterprise on Windows (QSEfW). You can configure Okta as an identity provider (IdP) for use with QSEoK and QSEfW.

You will create the following:

- an application for interactive login (QSEoK)
- programmatic use of Okta

Creating an Okta application and user for QSEoK for interactive logins

Create an Okta application and a user. An Okta application allows an application, (QSEfW/Qlik Cloud Services (QCS)/QSEoK), to use Okta for authentication.

We assume that you have an Okta account and tenant created.

When you install Qlik Sense Enterprise on Windows, with Multi-Cloud, you must use a developer account for Okta, see Okta Developer.

Creating a user

Create a user in Okta. You can skip this step if you have already created users.

Do the following:

1. Fill in first name and last name.
2. **Username**: Use your email address for user name.
3. Primary email: Same as **Username**.
4. For **Password**, select **Set by admin**.
5. Enter a password for the new user.
6. Optionally, clear the selection **User must change password in first login**.

Creating a new application in Okta

Create a new application, a tenant for QSEoK from Okta.

Do the following:

1. In Okta, go to **Applications** and click **Add Application**.
2. For **Platform**, select **Web** and click **Next**.
3. Enter a name for the app.
4. Enter a base URI.

   This is the IP address or server name from your QSEoK. Example: https://40.118.9.61
5. Enter a login redirect URI.
   As for the base URI, you use the IP address or server name from your environment. Example: *https://40.118.9.61/login/callback*
6. In the **Grant type allowed** section, for client acting on behalf of itself, select **Client Credentials**.
7. Click **Done**.

Configuration for programmatic access

Configure Okta to support usage programmatically (in this case to support distribution to QSEoK or QCS).

Creating an Okta API resource server and application for programmatic access

In Okta, you create a new Resource Server API. In this case, the Okta Resource Server API represents the protected QSEoK resource API. In OAuth terms, you need to configure Okta for the Client Credentials Grant flow.

First, create a new Authorization Server (under the API tab) for your tenant.

Do the following:

1. In the top menu, select **API**.
2. Open **Authorization Servers**.
3. Click **Add Authorization Server**.
4. Fill in name, audience (must be `qlik.api`), and description.
5. Save the API.
6. Open the **Scopes** tab.
7. Click **Add Scope** tab.
8. Enter a name and description, and select **Set as default scope**.
9. Click **Create**.
10. Open the **Access Policies** tab.
11. Click **Add Policy**.
12. For name and description, enter *Grant Clients*.
13. For **Assign to**, keep the selection **All clients**.
14. Click **Create Policy**.
15. Click **Add Rule**.
16. Enter a name for the rule.
17. Clear the selections under **Client acting on behalf of a user**.
18. Click **Create Rule**.

Creating an Okta application for programmatic authentication

Just like you created an Okta application for interactive logins above, you will now create an Okta application for programmatic authentication.

Do the following:

1. In the Okta top menu, open **Applications**.
2. Click **Add Application**.
3. For **Platform**, select **Service** and click **Next**.
4. Enter a name for the app.
5. Click **Done**.

Using Okta as an IdP for Qlik Sense Enterprise on Kubernetes

You can configure Qlik Sense Enterprise on Kubernetes (QSEoK) to use Okta as an identity provider.

After completing the steps, you will be able to log into a QSEoK tenant using an Okta user name and password as well as interact with the QSEoK tenant programmatically.

We assume that you are running QSEoK on a Mac which has Kubernetes running using Docker for Mac. Also without this exact configuration, you should be able to use the same concepts if running Kubernetes in other supported ways.

Configuring QSEoK to use Okta IdP

Before you start, make sure you have the following:

- Okta account
- Okta tenant
- Okta app, configured with interactive login and programmatic access.
- Configuration settings from your Okta application:
  - *discoveryUrl*: The OpenID Connect Discovery URL which allows applications, such as QSEoK, to use Okta with minimal configuration.
  - *clientId*: Uniquely identifies the client that is using Okta for authentication.
  - *clientSecret*: Secret that the client uses along with the Client ID to use Okta for authentication.

Many of the code examples contain placeholder values that need to be replaced by your own values.

You provide configuration to QSEoK by using a *values.yml* file. The *values.yml* file should look like the following example:

```
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>"
```

You need to enter the values for *discoveryUrl*, *clientId*, *clientSecret*, *realm*, and *hostname*.

In Okta, you can find your *Client ID* and *Client secret* under the **General** tab in the **Client Credentials** section for the application you created.

Applying the configuration to your cluster

Use Helm (see https://helm.sh/) to apply the configuration in your *values.yml* file to our Kubernetes cluster:

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

To make sure that your configuration has been applied you can run `get values` command to see the resolved configuration:

```
$ 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>"
```

Configuring your hosts file

This section is only relevant if there is no DNS.

For `<hostname>` to resolve, add the following to your `/etc/hosts` file:

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

Log in to your tenant

You are now set to log into your tenant. In your browser, go to *https://<tenant address>* and you should be redirected to an Okta login page. After a successful login you reach a home page to which apps are distributed.

Adding programmatic configuration to QSEoK

You now need an IdP configuration to QSEoK to point to the application and authorization server created above. Note that a `primary: true` was added to the existing configuration you had.

```
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"]
```

Use Helm to apply the configuration in your *values.yml* file to your Kubernetes cluster:

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

4.3 Configuring certificates in your Qlik Sense Enterprise on Kubernetes deployment

By default, Qlik Sense Enterprise on Kubernetes is installed with a self-signed certificate that will not be trusted by users browsers. To replace this with a SSL certificate that you own, complete the steps below.

In this example, the certificate is in a file called tls.crt and the associated private key is in a file called tls.key.

Create the secret resource in Kubernetes

1. Create a file called secret.yaml which will hold the certificate and its key. See the yaml definition below for an example:

```
apiVersion: v1
kind: Secret
metadata:
  name: my-certificate
  namespace: default
type: kubernetes.io/tls
data:
  tls.crt:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURORENDQWh3Q0NRRHUxeDdVdEdJSjJEQU5CZ2txaGtpRzl3MEJBUX
NGQURCY01Rc3dDUVlEVlFRR0V3SkQKUVRFUU1BNEdBMVVFQ0F3SFQyNTBZWEpwYnpFUE1BMEdBMVVFQnd3R1QzUjBZWGRo
TVJRd0VnWURWUVFLREF0TQpaWGhqYjNKd0lFbHVZekVVTUJJR0ExVUVBd3dMYkdWNFkyOXljQzVqYjIwd0hoY05NVGd3Tm
pJM01Ua3hOelV3CldoY05NVGt3TmpJM01Ua3hOelV3V2pCY01Rc3dDUVlEVlFRR0V3SkRRVEVRTUE0R0ExVUVDQXdIVDI1
MFlYSnAKYnpFUE1BMEdBMVVFQnd3R1QzUjBZWGRoTVJRd0VnWURWUVFLREF0TVpYaGpiM0p3SUVsdVl6RVVNQklHQTFVRQ
pBd3dMYkdWNFkyOXljQzVqYjIwd2dnRWlNQTBHQ1NxR1NJYjNEUUVCQVFVQUE0SUJEd0F3Z2dFSOFvSUJBUUQyCndTZk03
TDJiTHBnR1VsRERIci8rSUc3YlFONndnTzVMYWdqcnJramFHRjRGcVc0NS9Ha3hHTVhlZzZSTUpuYnkKTTI1RVloY2ZSVl
lzTmtaQVpCakYzM2ZwNlBqYjhydzhZV016RnJMOUtkeG8rZEtYSE14MTkvaExTaS82QTJOMgpBNzJta1krT2JmMHl0R1B5
aEZVY0lEZEFxbWtGTitoTXlGZjQwS0l5VS94NjZMVHhsYjZLQm1uZm9LK3VlNStZCmVxcGVLRkhBZkZwK1NFSG5UMkNJZX
dmQXR0Z29NL3dyREZVcENPS0sxZEJMUytzbzBZOUFCWG9wRm05U1RGV00KNVdZMno1NWdoa1l6UUpmemlRMC9WZkppamdM
WmhwcjRCRVAwaXV0dlRIczhrSFIyUTJqSkxVNEpncFViTnlXOQpUejFLQzhMVGpiL1J4RVJCK01FWkFnTUJBQUV3RFFZSk
```

tvWklodmNOQVFFTEJRQURnZ0VCQUlJSitsc2tlcUwzClpPVmxwNm1ZakRmRGt5Z1dkQVlCU0pINDdidHJkQUZrSlpQVVVM
Tzl6cTAxd0VBUHArRzFqZGUzSG42QmtyM0cKcklsRFZEckFobGRLR0hONS9BMXdoMjBxVlNKZzBkV25ielJnSVlkK0ljdj
hIa1RORGlUSDRxWHRjUHR2VHQ0NgphTmJUOHc2THl0Sm9YRWNZVEQzcjVXdERKWHFodkdHUjVLNU1Ubmo3QmZxcXNsS2M4
ZUZVbFdxOUpJZEZGZGRNClNzbHlhdE1zbk9YMGxtdDc4VnZYRFJ1RU5RM3BYUU1wNkhyQ3ZHRnY0NlhLRE5scVNyZm9vSU
RvbHBaUW5CdHAKWmdGZHc4U3VtMW91NVo1V0QxeW9XYkdLdFlLaTFiZkdNRHlsNURqZHdIakk0dG5GZk5LQ2E5TGZMdG5h
S2V2RwprcS8wOTBtcjJxcz0KLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQo=
  tls.key:
LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcFFJQkFBS0NBUUVBOXNFbnpPeTlteTZZQmxKUXd4Ni8vaU
J1MjBEZXNJRHVTMm9JNjY1STJoaGVCYWx1Ck9meHBNUmpGM29Pa1RDWjI4ak51UkdJWEgwVldMRFpHUUdRWXhkOTM2ZWo0
Mi9LOFBHRmpNeGF5L1NuY2FQblMKbHh6TWRmZjRTMG92K2dOamRnTzlwcEdQam0zOU1yUmo4b1JWSENBM1FLcHBCVGZvVE
1oWCtOQ2lNbFA4ZXVpMAo4WlcraWdacDM2Q3ZybnVmbUhxcVhpaFJ3SHhhZmtoQjUwOWdpSHNId0xiWUtEUDhLd3hWS1Fq
aWl0WFFTMHZyCktOR1BRQVY2S1JadlVreFZqT1ZtTnMrZVlJWkdNMENYODRrTlAxWHlZbzRDMllhYStBUkQ5SXJyYjB4N1
BKQjAKZGtOb3lTMU9DWUtWR3pjbHZVODlTZ3ZDMDQyLzBjUkVRZmpCR1FJREFRQUJBb0lCQVFEaW9FNU1vTllYazNpZQp2
YlZkTDMzSUNjT203WEpaaTJEUXRLZFN4alEwMHBKd0FzZXd6QWxVeFZyZDNldms0S0w0R1pKWmpmbU1oK0w0CklqVHRhTU
Z6NHFWQW1POFBHMllVMHFFSVIvM0dGRTlSdnJqU2Z1bXVJenZROG1jVDZVN05FZXg1OGxCMTBNRHUKYzgwajdMUTZhOEF6
VFErOWNqYVJacU9kYXdpOHpLWnVpdkhRazFXelU0eTZyYUhRSEliVnZOWHAxMExOK1RKYQpsRk83RlVwOG05bGh5UllqSy
9Vczd6Y0JOaWVRSWRTVGV6QkUvNzlwY0d4Vmd0Yk4xYTkyS2FjR0YzQWpzODF4CnNSQkc2Z2dMbTJ4U25Qdng2Y1ZvRkw4
VVNERUtOcDJBKzNnZlh0ZWg4RWNQSDNVcEZNT29YK0swRkh5bUxnY08KZ1g2R0U4OGRBb0dCQVA1MDZaRlcrTTRpbG1ibk
42VEFoeVpjbXY3RW1oVDJDb3NaeGUwbUYwcGE4YVUzWmhoOAorUk5nYk4rTjAwbWU3cmtmd2F6L2pMZUpxYUhibTdFSXdE
NWVQakRsR2lRMmh3L0ZPdzhCWkc5OFYzZVJBblQzCmE3UWxCVmRJcGRUaDVHYjlPbDhFbmN0Ti9ncXdaRi9Xa2wwSmF5VX
pHdzg0elFsMm4vRWg1MitqQW9HQkFQaEEKU0wzL3NCK3U2aDkzSUdBOFR4UURPQjJXMnFpY1g4aXkrd3p2U056Z1BCRTlT
ZTFSQytCMHNPZnV5NStGYUw1OApuazVPSnVjeWVuNXlkYTB1dndaV0ZFUWtkVGxtclRIV3F1d1ZCWWpad1RzVHBuTFp6TW
g1ZWE2TTEzV3ZRamdOClRGK1pENUd1QUZuaEpkaHlFbGg5NllpVXFvZkhCbVByYlVsMzNpZ1RBb0dCQU9scXFjOXZMckU1
UFNxTU01am8KRzVIdkpTZExoZVJ3aHQ5dTl3ZENCOTluYVgrYlFBWjZyYWsrck9DdG93Skh6c2oyL3AwSmx6WE9ERk96dG
pCQgp6TWZwZjdjdWtqcGg0cWR1ejJ4R1pMRjRLR3hBRXpia3VHSDZDOW96aEJ1eVUraTlwa3YvV2hoakpRVDlKalpjCkNN
dGJsblA3VzNrdEs3amdubnQrRkdWdEFvR0JBTHk5RXhEdzdsU0lSWk82bkRET2FVakwwY1FkUnd6ZUpxeU4KOXZTMGorN3
R4SDFPM1gvQ0dJZUQ4R1BGVjZabVpXWTlsSlh3TVAraGp5UEhuZFFYTUVCKy9WVjVhbTVEcy9XcQpRYlRQbnErVzdxUWRv
blM3UmtnOG90aWtWOE12aGViYnBXOGhEWWN5aFMzUVVWZW9FbTZZL0E1TkNRSEZ2UVFHCm44WTFqZjlsQW9HQUM2UFFYZ3
dwOTdudzMrWnZBeDJSUktqOHA2VUtraXVpRUlTQnM2ekR4c2hxdHJqR2VpaUQKRDNUNnBHVmhlYU94Y05reFRIVTlCcVo4
blNxd080cThjS2xLdXB0SkZFcFNtOVMrMWMveFdvNTdyWlVuNWlLbApDQzFWTkR6anYrRWNjU1FjOGJPZUxTRjd4cjRyYT
dnK3pRR0lSdXlvK005TDVBbjJxaEphM1ZRPQotLS0tLUVORCBSU0EgUFJJVkFURSBLRVktLS0tLQo=

2. You can give the name field a meaningful name. In this example we've use my-certificate.
   The tls.crt field is the base64 encoded value of your certificate. You can get this value using the following command:
   ```
   cat tls.crt | base64
   ```
3. The base64 decoded value will be displayed on the screen. Enter it for the tls.crt value in your .yaml file.
4. Do the same for the tls.key:
   ```
   cat tls.key | base64
   ```
5. Enter the resulting base64 value in your .yaml file.
6. Now create the secret resource in Kubernetes using the following command:
   ```
   kubectl apply -f secret.yaml
   ```
7. You can verify the secret has been created using the following command:
   ```
   kubectl get secret my-certificate
   ```

Configure the Ingress to use the Certificate

1. Configure the Qlik Sense ingress to use the secret created in the previous procedure by adding the following to your values.yaml file:
   ```
   # References the “my-certificate” secret created within the “default” namespace
   elastic-infra:
    nginx-ingress:
   ```

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

2. Update your cluster using the following command:
   `helm upgrade --install qliksense qlik/qliksense -f values.yaml`

Verifying the certificate with your browser

- Using your browser, go to the domain you configured to verify the certificate presented by Qlik Sense's ingress controller.

4.4 Configuring MongoDB in Qlik Sense Enterprise on Kubernetes

Qlik Sense Enterprise on Kubernetes uses **MongoDB** as a database for persisting content for several services (excluding Qlik Sense app files).

By default, a pre-configured **MongoDB Community Edition** is added during the installation of Qlik Sense Enterprise on Kubernetes (QSEoK). This is only intended to be used for quick start, testing and evaluation purposes. If you use this version, your MongoDB data may be lost if the Kubernetes cluster is updated.

You can set up a production-ready **MongoDB** environment in the following ways:

- Deploy a separate MongoDB server or cluster along-side Qlik Sense.
- Use a MongoDB DBaaS provider (such as **MongoDB Atlas** or **mlab**)

Configuring the MongoDB connection

When installing QSEoK you can specify your MongoDB connection as follows:

- A parameter in the `helm install` command.
- Referencing the connecting settings in a **values.yaml** and using this in the `helm install` command.

Using CLI paramaters

You can extend the basic `helm install` command by setting the following properties:

- Set the **devMode.enabled** value to `false` to disable development mode.
- Set the **mongodb.uri** value with the connection string to MongoDB.

Example:

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

Example: Connection URI format

The connection string format will vary depending on the configuration used. This format will be different if using a Kubernetes secret to connect with.

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

Some services assume that SSL is enabled in MongoDB. If SSL is not being used then add `?ssl=false` to end of the connection URI.

Referencing values.yaml

Create the **values.yaml** file and include the settings you want to reference in the `helm install` command.

- Set the **devMode.enabled** value to `false` to disable development mode.
- Set the **mongodb.uri** value with the connection string to MongoDB.

Example: values.yaml

```
engine:
  acceptEULA: "yes"

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

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

The values.yaml file is then referenced in the `helm install` command:

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

Connecting to MongoDB with SSL

You can set up an Secure Sockets Layer (SSL) connection to the MongoDB repository database.

Works on Qlik Sense Enterprise on Kubernetes June 2019 and later.

The MongoDB server must have been set with an SSL certificate and allow SSL connections.

Add `?ssl=true` to end of the connection URI to connect the MongoDB repository database with SSL.

```
devMode:
  enabled: false
```

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

If the MongoDB database is set up with a self-signed certificate or certificates not issued by a public Certificate Authority (CA), the CA certificate chain needs to be added in the **values.yaml** file:

```
global:
  certs:
    enabled: true
    configMap:
      create: true
      name: "{{ .Release.Name }}-ca-certs"
      certs: |+
        -----BEGIN CERTIFICATE-----
        MIIDLDCCAhSgAwIBAgIQANxWuceSqgA8h3fJ1Q7ZiTANBgkqhkiG9w0BAQsFADAm
        MSQwIgYDVQQDDBtRbGlrU2VydmVyMi5kb21haW4ubG9jYWwtQ0EwHhcNMTcwOTA5
        MTA1NjMwWhcNMjcwOTE2MTA1NjMwWjAmMSQwIgYDVQQDDBtRbGlrU2VydmVyMi5k
        b21haW4ubG9jYWwtQ0EwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCZ
        mnjqNBml1RU6vbRlakPNrCasFZSherijN4Rzj54BMb0cljC2ZfOnve2ZS5k27Dp9
        Yt/S3OB6MQTRNZBCOCOw3jqnOW87iemxhoE713EkF+zNcWnhHRA53+be1iIhV+kE
        fyFl6/4QQmUmZo2hu6gIajmdtZvM/CgjiPdF5a6KQp4WHA339afuIMR5KQe1Qt7E
        xqaTbh7niOJEXZXSHcbT8OsFam4O36rGMpjuseDbJgsIlLGSw1QwnIxWf+bFlbiD
        +2XJclAVXt+BCrsYfBXd3akLDulStW+X9SFFqX1V8+rkDdOV8lffaNhN6K4HQeBG
        ...
        LmIMzgUM9+baRYUwC552X6+szY55xqY21OyjFGSqrZDyyrJMi9RhDhSL1ZqIIJDm
        kpjNMY87Qa/c2slWTjg9lE/55OnBFZfQoD1zODVALCil9Tlb43wRsn8nMdD4U6Qz
        cgYfPkhRw2oUZuZwTmPOIYMrWPmmGXY4T9lZrq5afS7p+et1TKXZEZAC7akXDYL4
        CRjjXsfxmDaxy8sefg+LOnHgvESc1hWEBD2LlVVWbZCFi4MrwkkDyik5NWu6GkN2
        2xi+CJX3EBhHb1aFVDGd5dBSv3agXatsAnUMzxquuvtKbrURbMfyPCyiAZwlG9AN
        -----END CERTIFICATE-----
```

Reference

- **`enabled`**: enables (`true`) or disables (`false`) the usage of a global CA certificate.
- **`create`**: enables (`true`) or disables (`false`) the creation of a CA certificate configMap.
- **`name`**: the templated CA certificate configMap name based on the release name.
- **`certs`**: the global CA certificate chain. This replaces any existing CA trust chain.

4.5 Configuring a proxy for Qlik License Service communication in Qlik Sense Enterprise on Kubernetes

You can handle the communication between the Qlik License Service and the License Back-end with a proxy.

The Qlik License Service is included in Qlik Sense Enterprise February 2019 and later releases and is used when Qlik Sense is activated using a signed key license. The Qlik License Service stores the information

about the license, and communicates with a License Back-end Service, hosted by Qlik, for product activations and entitlement management. Port 443 is used for accessing the License Back-end Service and retrieving license information.

With Qlik Sense June 2019 or later you can configure the communication between Qlik License service and the Qlik License Back-end to be handled by a proxy.

In Qlik Sense Enterprise on Kubernetes, configuration of a proxy for the Qlik License Service is done using helm configurations. Both HTTP and HTTPS scheme are supported.

Add the following content in the *values.yaml* file under the `Licenses` section:

```
## 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:
```