opentext™

Kubernetes Deployment Guide

OpenText™ Intelligent Viewing

Deploy Intelligent Viewing services to a Kubernetes cluster.

CLIVSA250400-IGK-EN-01

Kubernetes Deployment Guide
OpenText™ Intelligent Viewing
CLIVSA250400-IGK-EN-01
Rev.: 2025-Oct-15

This documentation has been created for OpenText™ Intelligent Viewing CE 25.4.
It is also valid for subsequent software releases unless OpenText has made newer documentation available with the product, on an OpenText website, or by any other means.

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111
Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677
Support: https://support.opentext.com
For more information, visit https://www.opentext.com

Table of Contents

Part 1
Deployment

Chapter 1

Preface

This document describes how to deploy OpenText™ Intelligent Viewing (OTIV) services to a Kubernetes cluster using container images provided by OpenText.

OpenText Intelligent Viewing provides a cloud-designed collaboration viewer for extensive file formats as well as the ability to search text, annotate, redact, measure, print, and publish files from within the viewer.

For a full list of the document formats supported by Intelligent Viewing, see supported document types (https://www.opentext.com/assets/documents/en-US/pdf/opentext-intelligent-viewing-supported-formats-en.pdf).

OpenText provides helm charts and Docker container images on the OpenText helm and docker registries that can be used to deploy Intelligent Viewing in a Kubernetes cluster on a cloud platform such as Google Cloud Platform.

Basic instructions for the creation of a Kubernetes cluster on your organization's cloud platform are provided in *OpenText Content Management - Cloud Deployment Guide (SULCCD-IGD)*. For more detailed information, refer to your cloud provider's documentation.

Tip: Kubernetes is an open-source platform orchestration engine for automating deployment and management of Docker containers. It provides many features, including automatic scaling, high availability, and fault tolerance, and simplifies the deployment of Intelligent Viewing.

A visual application (Kubernetes Dashboard) is available for inspecting a Kubernetes cluster. For information on installing and invoking this dashboard, see https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/.

1.1 Integration documentation

After deployment of the Intelligent Viewing solution (see "Install OTIV Services" on page 25), information and tutorials about integrating to OpenText Intelligent Viewing can be found on the OpenText Developer network **Products > Viewing & Transformation Services** page:

https://developer.opentext.com/services/products/viewingtransformationservices

1.2 Intelligent Viewing Sample Application (IVSA)

A sample application that provides a basic interface to Intelligent Viewing functionality is externally available. For download, setup, and configuration instructions for the Intelligent Viewing Sample Application (IVSA), refer to:

https://github.com/opentext/ivsa

1.3 Glossary of terms

This glossary provides a brief explanation of some of the terms used in this document.

Cluster

A Kubernetes Cluster is composed of multiple Kubernetes Nodes that intelligently handle distributing work to the individual nodes that run containerized applications and tasks.

ConfigMap/Secrets

The configuration information that gets associated to pods.

Container

A Docker container is an instance of a Docker image.

Deployments

A controller that attempts to make sure a specified number of replicas for stateless pods are running.

Helm Chart

A Helm chart is a collection of files that describe a related set of Kubernetes resources. The name of the chart is the directory within which the files are stored. The deployment of Kubernetes resources is done using a helm chart. Helm chart configuration is done by updating the helm properties in a YAML file, or through command line arguments.

Image

A container image is built up from a series of layers. Each layer represents an instruction in the image's `Dockerfile`. Each layer except the last one is read-only. An image name is made up of slash-separated name components, optionally prefixed by a registry host name.

Ingress

Routes external http(s) traffic from an end-user machine to Kubernetes services within the cluster.

Namespace

A logical grouping of Kubernetes resources, typically representing an application.

Node

The smallest unit of computing hardware in Kubernetes representing a single machine in your cluster.

Pod

A Kubernetes pod is a group of one or more containers that are deployed together and running on the same node. Containers in a pod share resources and the local network.

PVC

Persistent Volume Claim is a persistent volume (similar to a disk) mounted by a pod.

Service

An internal Kubernetes route that connects a service name to a target pod's port.

Statefulset

A controller that attempts to make sure a specified number of replicas for stateful pods are running. Stateful pods retain data between restarts.

Storage class

A class of storage provided by the Kubernetes cluster. Classes can provide different performance capabilities and access modes (for example, `ReadWriteOnce/ReadWriteMany`).

Tag

Image Tags identify variants of container images. A single image can be given one or more tags. The combination of `repository:tag` can be used to identify the intended location of an image.

Chapter 2

Environment overview

OpenText Intelligent Viewing (OTIV) services require access to an OpenText™ Directory Services (OTDS) server for licensing and for OAuth 2 authentication. OTDS can be installed from the OTIV foundation helm chart, or the OTIV services can integrate to an OTDS installed using another product.

If deploying to Google Kubernetes Engine (GKE), refer to *OpenText Content Management - Cloud Deployment Guide (SULCCD-IGD)* for an overview on setting up the Google Cloud Platform (GCP) environment. For GCP installation requirements, see *OpenText Content Management - Cloud Deployment Guide (SULCCD-IGD)*. Note that an SSL certificate must be provided from your organization's infrastructure, and a wildcard DNS record should be created to account for the various OTIV services that must be externally accessible. Although recommended, a wildcard DNS entry is not a requirement and you can optionally use individual DNS entries for each service.

Several of the OTIV services store data in a PostgreSQL, Microsoft SQL Server, or Oracle database. Supported versions include:

- PostgreSQL versions 14 through 17
- Oracle version 19c (with the latest patch) through 21c
- Microsoft SQL Server 2019 and 2022

2.1 Intelligent Viewing container images

The following docker container images are used to deploy Intelligent Viewing. These images and tags (current version) are listed in the `imageTags.yaml` file included in the Intelligent Viewing Helm chart.

OpenText Directory Services
Image name: `otds-server`

OpenText Intelligent Viewing Asset Service
Image name: `otiv-asset`

OpenText Intelligent Viewing Configuration Service
Image name: `otiv-config`

OpenText Intelligent Viewing Highlight Service
Image name: `otiv-highlight`

OpenText Intelligent Viewing Markup Service
Image name: `otiv-markup`

OpenText Intelligent Viewing Publication Service
Image name: `otiv-publication`

OpenText Intelligent Viewing Publishing Agent
Image name: `otiv-publisher`

OpenText Intelligent Viewing Viewer Service
Image name: `otiv-viewer`

OpenText Intelligent AMQP Service
Image name: `otiv-amqp`

2.2 Installation order

Recommended installation sequence.

- PostgreSQL. Microsoft SQL Server, or Oracle instance on a VM
- Otiv Helm Chart - installs OTIV services

2.3 PostgreSQL setup notes

While the PostgreSQL docker container can be installed as a sub-chart of the of the OTIV helm chart, for production deployments, it's required to install PostgreSQL external to the Kubernetes instance in a VM.

For production deployments, it's required that PostgreSQL not be deployed as container due to the possibility of it being evicted or otherwise restarted by the Kubernetes environment. It should be installed on a stand-alone system such as a VM. Note that `bitnami/postgresql` is a sub-chart of the OTIV helm chart and can be deployed as a container for development or testing purposes.

2.4 Oracle setup notes

If Oracle is your target database, do the following.

Upgrade note:

If you are migrating an Intelligent Viewing installation at version 24.2 or earlier to a more recent version, the Oracle database schema has changed starting in 24.3 and you must run the Oracle SQL script to migrate the data from the previous schema to the new schema.

This schema update script should be run independent of any other scripts that are executed during database maintenance. Because the script provided contains a COMMIT statement within it, if you are running any other scripts in succession, those scripts will also get committed into the database.

For more information, see this OpenText support article (https://support.opentext.com/csm?id=kb_article_view&sysparm_article=KB0811280).

Follow the steps below to create a Pluggable Database (PDB) in the Container Database (CDB):

1. Open **sqlplus** from a command line:

```
sqlplus '/ as sysdba'
```

2. Verify that you are currently in the CDB root:

```
SHOW CON_NAME;
CON_NAME
------------------------------
CDB$ROOT
```

3. Create a pluggable database:

```
CREATE PLUGGABLE DATABASE <pdb_name> ADMIN USER <pdb_admin_username> IDENTIFIED BY
<pd_password> ROLES=(DBA) FILE_NAME_CONVERT = ('<path to pbdseed directory>\pdbseed
\', '/<pdb_name>/');
```

4. Connect to the pluggable database:

```
CONNECT <pdb_name> as sysdba;
password: <type_the_pdb_password>
ALTER SESSION SET CONTAINER = <pdb_name>;
```

5. Verify that the current connection is the newly created PDB:

```
SHOW CON_NAME;
```

6. Startup the PDB if it is not already open/started:

```
STARTUP;
```

7. Create a tablespace against the newly created PDB::

```
CREATE TABLESPACE iv_ts_space DATAFILE 'iv_ts_space_01.dbf' SIZE 10M AUTOEXTEND ON
MAXSIZE UNLIMITED;
```

Note: If you want to use the internal OTDS pod, then check for the USERS tablespace existence:

- You can check whether Oracle has already created the USERS tablespace by default using the following command:

```
SELECT tablespace_name FROM dba_tablespaces WHERE tablespace_name = 'USERS';
```

- If this command returns zero records, or the USERS tablespace is not available, create the tablespace using the following command:

```
CREATE TABLESPACE USERS DATAFILE 'USERS_01.dbf' SIZE 10M AUTOEXTEND ON MAXSIZE
UNLIMITED;
```

8. Create a user for the tablespace:

```
CREATE USER <username> IDENTIFIED BY <user_password> DEFAULT TABLESPACE iv_ts_space
TEMPORARY TABLESPACE TEMP;
ALTER USER <username> QUOTA UNLIMITED ON iv_ts_space;
GRANT CREATE TRIGGER,CREATE TABLE,CREATE PROCEDURE,CREATE SESSION TO <username>;
GRANT SELECT_CATALOG_ROLE TO TO <username>;
```

Notes

- Set the helm properties `global.database.adminUsername` and `global.database.ivUsername` with the above <username> value. Set the helm properties `global.database.adminPassword` and `global.database.ivPassword` with the above <user_password> value.

9. If you have already created the PDB as a test and want to restart from scratch, use the below commands to drop the pluggable database:

```
sqlplus '/ as sysdba'
ALTER PLUGGABLE DATABASE <pdb_name> CLOSE INSTANCES=ALL;
DROP PLUGGABLE DATABASE <pdb_name> INCLUDING DATAFILES;
```

2.5 Microsoft SQL Server setup notes

To configure MS SQL database support, refer to the `global.database` helm properties described in "Edit the helm installation parameters" on page 19.

Chapter 3

Installation prerequisites

This section describes the installation requirements necessary to support OpenText Intelligent Viewing.

3.1 Register domain and create SSL wildcard certificate

To enable public access to your site, you must register your domain if not already done. If hosting on AWS, this can be done through its `Route 53` service. To allow https communication to Intelligent Viewing services, you must create an SSL wildcard certificate for the sub-domains of the registered domain.

On AWS, the SSL wildcard certificate can be created through its `Certificate Manager` service. Otherwise, it must be imported. After the certificate has been issued or imported, its value will be populated in the ARN field. This value will be referenced in an ingress annotation helm property value (`global.ingressAnnotations`), described in "Edit the helm installation parameters" on page 19.

3.2 Kubernetes cluster requirements

- In Amazon's Elastic Kubernetes Service (EKS), you must install the application load balancer (ALB) ingress controller. In all other environments, the Ingress NGINX Controller has been certified to handle network traffic to and from your product's deployment.

  Tip: For information on installing the ALB Ingress Controller, see https://aws.amazon.com/blogs/opensource/kubernetes-ingress-aws-alb-ingress-controller/.

  For information on installing the Ingress NGINX controller using helm, see https://github.com/kubernetes/ingress-nginx/tree/master/charts/ingress-nginx.

- If installing the Ingress NGINX Controller, execute the command:

  ```
  kubectl -n <nginx-namsespace> services
  ```

  Make note of the external IP address. Add a wildcard DNS type A record for the sub-domains of the registered domain that points to the External IP address (for example, *.registereddomain.com > {External-Ip}). Note that DNS records for the AWS EKS environment using the ALB Ingress Controller are established after Intelligent Viewing is deployed.

- Storage class that has ReadWriteMany (RWX) access. This is typically provided by a NFS storage class provisioner. For more information, see https://

medium.com/asl19-developers/create-readwritemany-persistentvolumeclaims-on-your-kubernetes-cluster-3a8db51f98e3.

- Docker containers deployed to the cluster need network access to the externally installed PostgreSQL instance, as well as OTDS if it is installed externally to this deployment.
- If installing on an OpenShift cluster, you must add one or more policies to enable Intelligent Viewing containers to run.

  To add a security context constraint (SCC) such as `nonroot` to the service accounts that are associated with the Intelligent Viewing containers, execute the following commands:

```
oc adm policy add-scc-to-user nonroot -z otiv-sa -n <namespace>
oc adm policy add-scc-to-user nonroot -z otiv-amqp -n <namespace>
oc adm policy add-scc-to-user nonroot -z otiv-asecret-sa -n <namespace>
oc adm policy add-scc-to-user nonroot -z otiv-pvc-sa -n <namespace>
```

  A more permissive alternative to the above is to issue the following command:

```
oc adm policy add-scc-to-group anyuid system:authenticated
```

3.3 Install machine requirements

- Access to the target Kubernetes cluster.
- Helm version 3.3 or later.
- Download and extract the CE 25.4 OTIV helm chart package. See "Download the Intelligent Viewing helm chart" on page 17.

For information regarding helm and its supported Kubernetes version, see https://helm.sh/docs/topics/version_skew/.

3.4 Other requirements

- The Intelligent Viewing deployment requires a Kubernetes secret object to enable encrypted communication between your deployment and the users who log on over the public Internet. The Kubernetes secret contains your My Support credentials to access the `registry.opentext.com` docker registry. Click here (https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry/#create-a-secret-by-providing-credentials-on-the-command-line) for information on creating this secret.
- If NGINX is your ingress controller, create a Kubernetes secret that contains your SSL certificate and private key information. For example:

```
kubectl create secret tls iv-secret --cert fullchain.pem --key privkey.pem
```

  If using the ALB Ingress Controller with EKS, the SSL certificate will be referenced through its ARN value as described in "Register domain and create SSL wildcard certificate" on page 15.

3.5 Download the Intelligent Viewing helm chart

The following download commands are used with the IV helm chart:

- This is a one–time setup and adds the opentext local helm repo to your list of repos:

```
helm repo add opentext  https://registry.opentext.com/helm --username <myuser
emailaddress> --password <mypassword>
```

- This step might need to run periodically. It refreshes your local list of available helm charts when a new chart or version is added:

```
helm repo update
```

- Use this step if you want to download the helm chart locally:

```
helm pull opentext/otiv --version 25.4.0
```

- This step extracts the pertinent files and directories from the downloaded chart. The `otiv/values.yaml` contains the helm installation properties.

```
tar xvfz otiv-25.4.0.tgz otiv/README.md otiv/values.yaml otiv/sizings otiv/scripts
```

Chapter 4

Installation steps

This section describes the steps for installing and configuring the Intelligent Viewing solution within a Kubernetes cluster. The remaining steps should be done from the directory where the OTIV helm chart package was extracted.

Intelligent Viewing can be installed as a high availability solution. For information, see *OpenText Intelligent Viewing - High Availability Install Guide (CLIVSA-IHA)*.

4.1 Edit the helm installation parameters

To prepare for the deployment of Intelligent Viewing, edit the YAML files listed in this section. The file `values.yaml` exposes properties for the OTIV helm chart (`otiv-<version>.tgz`). The `<size>.yaml` files within the sizings subdirectory contain artifact PVC size and resource sizing properties for different installation sizes. Review and edit these YAML files for your target deployment. See "Edit the helm sizing parameters" on page 24. Properties to pay attention to within `values.yaml` include the following. These property values are set based on your organization's specific requirements.

SSL support note

SSL support is only available for the PostgreSQL and MS SQL database.

`otdsTenantID`
If OTDS has been configured for multi-tenancy, specify the relevant tenant ID.

`otdsAdminUser`
Specifies the OTDS admin user name; defaults to "admin".

`singleCaCert`
Relevant if `global.enableOtivCustomizedTruststore` is `true`. Used to specify a single custom certificate to be used by OTIV services to communicate to services through SSL. If `global.enableOtivCustomizedTruststore` is `true` and this property is not set, the chart will look to import certificates from its `otiv/certificates` directory. Note that this property can be set through the command line with the `--set-file` argument. For example, `--set-file singleCaCert=myCert.crt`.

global.enableOtivCustomizedTruststore

When set to `true`, enables the import of custom certificates for use by the OTIV containers to communicate using SSL to services whose private certificates are not signed by a root certificate authority. Certificates to import must be placed in the helm chart's `otiv/certificates` directory or specified using the `singleCaCert` property (see above). Placing files in the `otiv/certificates` directory requires extracting the chart's contents (for example, `tar xvfz otiv-24.4.0.tgz`). The `./otiv` directory would then be used as the helm install/upgrade chart argument instead of the .tgz file.

Note: Not all encrypted communication requires the setting of this property. For example, this does not need to be set to true for connections to the `Azure Database for PostgresSQL flexible server`.

global.imageSource

Set to the registry where the OTIV docker images will be pulled from, typically: `registry.opentext.com`

global.ingressDomainName

Set to the domain name where the OTIV services will be running.

global.ingressIncludeNamespace

Determines if the ingress FQDN's hostname includes the namespace. For example, `otiv-asset-<namespace>.sample.org` where `sample.org` is set in the `global.ingressDomainName` property.

global.ingressOtiv<service>

```
global.ingressOtivAsset
global.ingressOtivHighlight
global.ingressOtivMarkup
global.ingressOtivPublication
global.ingressOtivViewer
```

These fields determine how the publicly accessible Intelligent Viewing FQDNs are set.

- If no value is set, the FQDN is set to `otiv-<service>-<namespace>.<ingressDomainName>` where service equals asset, highlight, markup, publication, and viewer.
- If the value contains at least one "." (such as `myasset.sample.org`), then the value represents the full FQDN of the Intelligent Viewing service and the namespace will not be included in the service URL (even though `global.ingressIncludeNamespace` value is set to true).
- If a value is set but doesn't contain a ".", the value represents the first DNS field of the FQDN, which is prepended to `ingressDomainName`. For example, `<ingressOtivAsset>-<namespace>.<ingressDomainName>`.

Note: In the above examples, `-<namespace>` is included only if `ingressIncludeNamespace` is set to `true`.

`global.ingressClass`
For AWS/EKS deployments using the ALB Ingress Controller, change this property to `alb`. Otherwise leave the value as `nginx`.

`global.ingressSSLSecret`
If the ingress class `nginx`, set this property to the TLS secret previously created. See "Other requirements" on page 16.

`global.ingressAnnotations`
For AWS/EKS deployments using the ALB Ingress Controller, change the subsequent lines to:

`alb.ingress.kubernetes.io/scheme: internet-facing`

`alb.ingress.kubernetes.io/certificate-arn:` <ARN for certificate established in "Register domain and create SSL wildcard certificate" on page 15 >.

`global.storageClassName`
Set to the storage class to be used by the `otiv-amqp` (rabbitmq) instance; this storage class just needs to provide ReadWriteOnce (RWO) access. The Intelligent Viewing Helm deployment uses the default storage classes in your Kubernetes cluster. If the default storage classes are acceptable, there is no need to change this information. To specify storage classes that are different from the default classes, enter a value as necessary.

For Azure Kubernetes Service (AKS), choose an **Azure Disk** storage class. To see This typically is the default storage class, or a storage class that starts with "managed".

`global.storageClassNameNFS`
Set to a storage class that provides ReadWriteMany (RWX) access. Intelligent Viewing requires a storage class with `RWX` access, which is typically implemented as an NFS storage class.

To list available storage classes for your k8s cluster, use:

```
kubectl get sc
```

For Azure Kubernetes Service (AKS), the value for this setting will typically be either `azurefile` or `azurefile-premium`.

`global.existingSecret`
Passwords used by the OTIV can be set in a Kubernetes secret prior to deployment to eliminate the need to specify them in the helm install or upgrade command. If this is done, this property should be set to the name of the Kubernetes secret containing the passwords.

See "Kubernetes secret integration" on page 29 for more information.

`global.imagePullSecret`
Set to the secret created within your namespace that provides OT Container Registry credentials (see prerequisites).

global.enableForwarding
If set to `true`, information in the request's Forwarded header will be used when setting the externally accessible URLs. Default is `false`.

otds.enabled
If you are integrating to an existing external OTDS instance, then set this property to `false`.

otds.otdsws.cryptkey
Used by OTDS for secure synchronization access to the back end database. The value must be set to a 16 character ASCII string that is base64 encoded (for example: Z2hkN2hyNDBkbWNGcVQ0TA==). This property is only relevant if OTDS is being deployed as part of this chart (`otds.enabled=true`).

global.otdsPublicUrl
If you are integrating to an existing external OTDS instance, then set this property to the URL of the external OTDS instance in the below format:

```
otdsPublicUrl: http(s)://<OTDSFQDN>:<PORT>
```

See also the multi-tenant note at the end of this section.

postgresql.enabled
If you are integrating to an existing external (VM based) PostgreSQL, SQL Server, or Oracle instance, then set this property to `false` and update the parameters `global.database.type` (with `postgresql`, `mssql`, or `oracle`), `global.database.hostname`, `global.database.adminUsername`, `global.database.adminPassword`, `global.database.port`, `global.database.adminDatabase`, and `global.database.ivName`.

global.database.type
Defaults to `postgresql`. If you are integrating to an existing external Oracle instance (VM based), then set this property value to `oracle`.

If connecting to a Microsoft SQL Server instance, set this property value to `mssql`.

global.database.hostname
Set to the Kubernetes service, the external IP, or the fully qualified domain name of the PostgreSQL, Microsoft SQL Server, or Oracle instance.

global.database.port
Set to the database port number. Defaults to `5432` for PostgreSQL. Typical port value is 1521 for Oracle and 1433 for Microsoft SQL Server.

global.database.ivName
Defaults to `otiv` for PostgreSQL.

global.database.adminDatabase
Set to the admin database.

- Defaults to `postgres` for PostgreSQL, which is the initial database that the database admin user connects to during docker initialization prior to the database defined in `global.database.ivName` being created.

- For Oracle, it corresponds to the name of the pluggable database created. See step #3 in "Oracle setup notes" on page 12.
- For Microsoft SQL Server, this will typically be set to `master`.

`global.database.adminUsername`
Set to the name of the PostgreSQL, Oracle, or Microsoft SQL Server admin user with appropriate (DBA for example) privileges.

- Defaults to `postgres` for PostgreSQL.
- For Oracle, this corresponds to `<pdb_admin_username>`, as set in Step 3 of "Oracle setup notes" on page 12.
- For Microsoft SQL Server, this will typically be set to `sa`.

`global.database.adminPassword`
Set to the password of the database admin user. For Oracle, this corresponds to `<pd_password>`, as set in Step 3 of "Oracle setup notes" on page 12.

`global.database.ivUsername`
Set to the name of the database user that connects to the database from the otiv application services. Defaults to `otiv`.

`global.database.ivPassword`
Set to the password of the application database user.

`postgresql.auth.existingSecret`
If `postgresql.enabled` is set to TRUE for development or testing purposes, and `global.existingSecret` has been set, then set this property's value to be equal to the value of the `global.existingSecret` property.

`global.database.ssl`
Set to `true` if your database is configured for SSL communication. For instance, an Azure Flexible Server integration for PostgreSQL requires SSL database communication.

`global.database.sslMode`
If `global.database.ssl` is set to `true`, setting this value will further moderate how the database SSL connection is established if the `global.database.type` is set to `postgresql`. Valid values are one of the following (listed in ascending order of security): `prefer`, `require`, `verify-ca`, or `verify-full`.

Defaults to `prefer`. For connections to the `Azure Database for PostgresSQL flexible server`, this should be set to either `prefer` or `require`.

`global.masterPassword`
Set to the OTDS admin password.

`global.amqp.password`
Sets the rabbitmq password.

otds.otdsws.otdsdb.password
Sets the database password for the OTDS user; the name of the OTDS user defaults to `otds`. This property is only relevant if OTDS is being deployed as part of this chart (`otds.enabled=true`).

Notes

- **Password notes**
  - If upgrading the helm chart, the value of the password properties must be set to the currently deployed values.
  - Changing the password of a component (for example, OTDS) must be done through the component UI or its external interface (such as psql for the database). After doing so, the helm upgrade can be run with the updated password(s).
- **Multi-tenant OTDS environments**

  When installing Intelligent Viewing within a multi-tenant OTDS environment, the following configuration changes must be made:

  1. Update the `otdsPublicUrl` property to the tenant-specific URL using the following syntax:
     ```
     otdsPublicUrl: http(s)://<OTDSFQDN>:<PORT>
     ```
  2. Update the `otdsTenantID` and `otdsAdminUser` values in the `values.yaml` file or run the helm command by passing these additional parameters:
     ```
     --set otdsTenantID=<TENANT-ID> --set otdsAdminUser=<TENANT-ADMIN-USERNAME>
     ```
- **Vault properties**

  For information about using the HashiCorp Enterprise Vault helm properties, see "Intelligent Viewing setup" on page 31.

4.1.1 Edit the helm sizing parameters

Determine which of the sizing files located within the chart's `otiv/sizing` subdirectory (`small.yaml`, `small-medium.yaml`, `medium.yaml`, `medium-large.yaml`, `large.yaml` or `extra-large.yaml`) to use for your target deployment, and adjust values as appropriate within that YAML file. For more information, see "Application scaling guidelines" on page 27.

4.2 Install OTIV Services

The following command (substituting for <*namespace*>) installs the OTIV services:

```
helm install -n <namespace> otiv ./otiv-25.4.0.tgz -f otiv/values.yaml -f otiv/sizings/
small.yaml
```

Note: Consider removing the `-f otiv/sizings/small.yaml` argument in development environments to reduce the amount of resources deployed to your Kubernetes cluster.

The helm command outputs the URLS of the externally accessible OTIV services.

4.2.1 Add DNS records for EKS deployments

If deploying to AWS EKS with the ALB Ingress Controller, you can now add DNS records.

Adding DNS records:

1. Execute the command:

   ```
   kubectl -n <namespace> get ingress
   ```

   The output will look similar to the following example:

   ```
   NAME
   HOSTS                                                                    ADDRESS
   otiv              otiv-asset.otiv.us,otiv-highlight.otiv.us + 5 more... 01c-ad-
   otivasset-889.us-east-2.elb.amazonaws.com
   ```

   Note: The list of hosts is truncated in the above command. The complete list of hosts can be displayed with the command:

   ```
   kubectl -n <namespace> get ingress -o yaml | grep host:
   ```

2. In AWS's Route 53 service UI, create DNS CNAME records for every host, mapping each to the value in the ADDRESS column from the above command.

4.3 Review the deployment installation

The status of the installation can be checked in the Kubernetes dashboard for the specified namespace, or by using the following command (substituting appropriately for <*namespace*>):

```
kubectl get pods -n <namespace>
```

The response should be:

```
NAME                READY           STATUS
otiv-amqp-0         1/1             Running
otiv-asset-*        1/1             Running
otiv-config-*       1/1             Running
otiv-highlight-*    1/1             Running
otiv-markup-*       1/1             Running
```

```
otiv-publication-*  1/1             Running
otiv-publisher-*    1/1             Running
otiv-viewer-*       1/1             Running
otdsws-*            1/1             Running
```

If the status reports something other than “Running”, the following commands can be run to obtain further information about the issue:

Status	Run command
ErrImagePull /ImagePullBackOff	`kubectl -n <namespace> describe pod <pod-name>`
CrashLoopBackOff	`kubectl -n <namespace> logs <pod-name>`
Pending	`kubectl -n <namespace> get pvc`

The `otiv-bas` PVC should have a status of `Bound`. A `Pending` status can mean that the storage class specified in `global.storageClassNameNFS` does not allow RWX access.

If problematic pod states persist, refer to troubleshooting tips listed in “Pod status troubleshooting” on page 45.

Chapter 5

Application scaling guidelines

When you deploy an application using Helm and Kubernetes, you define how many replicas of the application you'd like to run. To scale an application, you increase or decrease the number of replicas.

The YAML files within the `sizings` subdirectory `small.yaml`, `medium.yaml`, and `large.yaml` provide general resource sizing for different deployment sizes. Values in these files can be adjusted for your deployment needs. This needs to be done prior to running the helm install or helm upgrade command for the OTIV services.

Alternatively, to manually scale the number of pods for an IV service, enter the command:

```
kubectl scale deployment/otiv-<serviceName> --replicas=<replicaCount>
```

The relevant Intelligent Viewing services names are:

- asset
- config
- highlight
- markup
- publication
- publisher
- viewer
- amqp †

To scale the number of pods, increase the number of replicas. For example, to scale the number of publishing agent pods to 3, enter the command:

```
kubectl scale deployment/otiv-publisher --replicas=3
```

† The commands for scaling rabbitmq is slightly different. To scale the number of rabbitmq pods to 2, enter the command:

```
kubectl scale sts/otiv-amqp --replicas=2
```

It will take a few minutes for scaling to complete. When completed, you will see two instances of the `otiv-amqp` stateful set when you run `kubectl get pods`.

Chapter 6

Secret management

As an alternative to specifying password values directly during a helm install or upgrade command the password values can be set in a secret manager. The Intelligent Viewing helm installation supports different secret managers, such as Kubernetes Secrets and HashiCorp Vault. Other hyperscaler vault stores such as AWS Secrets Manager and Azure Key Vault can be supported indirectly through the use of the Secrets Store CSI Driver.

For example, in the case of Azure Key Vault, the Secrets Store CSI Driver can be configured to project the credential information from the Key Vault to a Kubernetes secret in the namespace where Intelligent Viewing is deployed. The Intelligent Viewing services can then be configured to retrieve the credential information from that Kubernetes secret; for more information, see "Intelligent Viewing setup" on page 30. Note that the Intelligent Viewing helm chart does not provide direct support for integrating the Secrets Store CSI driver for vault providers like Azure Key Store, so it's relying on this setup to typically be provided prior to its deployment, for instance in a parent helm chart that has included Intelligent Viewing as a subchart. For information on how a parent chart can integrate Azure Key Store with Kubernetes secrets, see **Use the Azure Key Vault provider for Secrets Store CSI Driver in an Azure Kubernetes Service (AKS) cluster**, which is in Microsoft Ignite as part of the Azure documentation.

Note that the Intelligent Viewing does provide some integration support for AWS Secrets Manager. For more information, see "AWS Secrets Manager integration" on page 32.

6.1 Kubernetes secret integration

A Kubernetes secret is an object that holds sensitive information, such as authentication keys, tokens, usernames, and passwords that can be used as configuration for pods running in your Kubernetes cluster. The secret is external to the pod itself, and is less vulnerable to being exposed in logs, screen shares, and other incidental leaks.

6.1.1 Kubernetes setup

The OTIV helm chart zip file contains an `example-secrets.yaml` file that demonstrates how to create a Kubernetes secret populated with the passwords and secret values used by Intelligent Viewing. After the YAML file (or a copy of it) has been updated with the appropriate values, a secret can be created within the specified Kubernetes namespace by executing the following command:

```
kubectl -n <namespace> apply -f example-secret.yaml
```

This creates a Kubernetes secret with the name provided for the YAML `metadata.name` property; for `example-secrets.yaml` this defaults to `otiv-secrets`.

6.1.2 Intelligent Viewing setup

When the helm install or upgrade command is run, the property `global.existingSecret` should be set to the name of the Kubernetes secret containing the values Intelligent Viewing needs. For example if using the default secret name provided by `example-secrets.yaml`, the helm property `global.existingSecret` would be set to `otiv-secrets`.

Note: If you are running PostgreSQL as a container for developing or testing purposes, also set the helm property `postgresql.auth.existingSecret` to the name of the secret.

As noted in the previous section, if upgrading the helm chart, the value of the password properties must be set to the currently deployed values.

6.2 HashiCorp Vault Enterprise integration

HashiCorp Vault is an identity-based secrets and encryption management system. Secrets in Vault are like Kubernetes secrets, but Vault provides encryption services that are gated by authentication and authorization methods. This section covers updates required for the Intelligent Viewing pods to integrate with HashiCorp Vault. It does not cover OTDS integration with HashiCorp Vault; reference the `otdsws.vault` properties within the OTDS chart for guidance on its Vault configuration.

6.2.1 Vault setup

The Vault administrator should configure K8s as the OpenID Connect protocol (OIDC) provider so that Vault can validate its service account token using jwt/oidc auth. For information about configuring K8s as the OIDC provider, see: https://developer.hashicorp.com/vault/docs/auth/jwt/oidc-providers/kubernetes

Note: Intelligent Viewing uses namespace concepts, which are available only in the Vault Enterprise product

The JWT authentication will be associated with a service account. The Intelligent Viewing service account is set by the helm property `global.serviceAccountName`

with a default value of `otiv-sa`. Use the value of `global.serviceAccountName` for the `bound_subject` when setting up the JWT authentication role.

Before installation of Intelligent Viewing, create a JSON secret in Vault using the following format:

```
{
  "ADMIN_USER_PASSWORD": "*********",
  "OTIV_DB_ADMIN_PASSWORD": "*********",
  "OTIV_DB_PASSWORD": "*********",
  "OTIV_HIGHLIGHT_CLIENT_SECRET": "*********",
  "OTIV_MONITOR_CLIENT_SECRET": "**********",
  "OTIV_PUBLICATION_CLIENT_SECRET": "************",
  "OTIV_PUBLISHER_CLIENT_SECRET": "************",
  "rabbitmq-password": "******************"
}
```

The names of the secret values can be configured, and not all values are mandatory. See the next section, "Intelligent Viewing setup" on page 31 for details.

6.2.2 Intelligent Viewing setup

The following helm properties are relevant for configuring integration with the HashiCorp Vault:

global.secretlink.enabled
Set to `true` to enable Vault integration. This creates a SecretLink side car that proxies communication between the Intelligent Viewing pods and the Hashicorp Vault.

global.secretlink.vault.address
Set to the URL of the Vault instance (for example `https://vault.example.com/`).

global.secretlink.vault.mountpoint
Set to the secret mount point (for example `kv`).

global.secretlink.vault.path
Set to the Vault secret's path (for example `/iv`).

global.secretlink.vault.namespace
Set to the Vault namespace (for example `iv`).

global.secretlink.vault.authpath
Set to the path to verify authentication established in "Vault setup" on page 30 (for example `auth/jwt`).

global.secretlink.vault.role
Set to the role created in Vault for IV (for example `iv-role`).

An example of the keys to set within the HashiCorp Vault were provided in "Vault setup" on page 30. These are the default keys that the Intelligent Viewing chart assumes are set. We recommend that you use the default key names for ease of troubleshooting with customer support, however, it is possible to adjust them to work within an existing Vault installation that has a pre-defined naming convention. The following helm properties are relevant if the Vault keys are changed:

global.otdsSecretKey
Defaults to `ADMIN_USER_PASSWORD`. Key for the OTDS admin password.

global.dbSecretKey
Defaults to `OTIV_DB_ADMIN_PASSWORD`. Key for the database administrator password.

highlight.clientSecretKey
Defaults to `OTIV_HIGHLIGHT_CLIENT_SECRET`. Key for the highlight service's OAuth client secret.

publication.monitoring.clientSecretKey
Defaults to `OTIV_MONITOR_CLIENT_SECRET`. Key for the OAuth client secret used for monitoring.

publication.clientSecretKey
Defaults to `OTIV_PUBLICATION_CLIENT_SECRET`. Key for the publication service's OAuth client secret.

publisher.clientSecretKey
Defaults to `OTIV_PUBLISHER_CLIENT_SECRET`. Key for the publishing agent's OAuth client secret.

global.amqp.pwdKey
Defaults to `rabbitmq-password`. Key for the RabbitMQ broker password.

global.ivDbSecretKey
Defaults to `OTIV_DB_PASSWORD`. Key for the database user password.

6.3 AWS Secrets Manager integration

AWS Secrets Manager is a secure vault used for securing sensitive data such as database credentials and OAuth secret values. It is possible for services running in Kubernetes pods to access information in the AWS Secrets Manager by using AWS Secrets and Configuration Provider (ASCP). For setup information, see https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_ascp_irsa.html.

The OTIV chart provides properties that work with ASCP to map one or more AWS Secrets Manager secrets to a Kubernetes secret that is used by the OTIV pods. Note that if a parent chart already provides the mapping from AWS Secrets Manager secret(s) to a single Kubernetess secret that contains all required Intelligent Viewing secrets, this AWS secrets mapping functionality is not required to be enabled within the OTIV chart. Instead, the `global.existingSecret` or OTIV's `existingSecret` property can be set to the name of this parent-created Kubernetes secret.

There are numerous helm properties under `global.otivSecretProviderClass` that are used to configure the integration with AWS Secrets Manager, including:

global.otivSecretProviderClass.enabled
Set to `true` to enable mapping of AWS Secrets Manager secrets to a Kubernetes secret.

global.otivSecretProviderClass.saRole
Set to the ARN of the IAM role that allows access to the AWS secrets.

Additional `global.otivSecretProviderClass` properties control mapping of the AWS Secrets Manager secrets to a Kubernetes secret and are discussed in the following section.

6.3.1 Mapping AWS secrets

The following seven Intelligent Viewing secrets must be defined in AWS Secrets Manger prior to deployment. It is assumed that OTDS will not be installed as part of the OTIV chart, so this list does not include OTDS specific secrets. Additionally, trust store certificates used to communicate to OTDS or the database through SSL must be set up in AWS Secrets Manager. For more information, see "Certificates" on page 34.

- Database Admin password
- OTDS Admin password
- RabbitMQ password
- Publication OAuth client secret
- Monitoring OAuth client secret
- Publisher OAuth client secret
- Highlight OAuth client secret

The RabbitMQ password and four OAuth client secrets must be setup within the same AWS Secrets Manager secret. The database and OTDS passwords can also be included this AWS Secrets Manager secret, or they can be placed in one or two separate AWS Secrets Manager secrets. Each of these secrets are stored in a separate key within their AWS Secrets Manager secret.

The OTIV chart provides four "objectName" helm properties, located beneath `global.otivSecretProviderClass`, that map the Amazon Resource Name (ARN) of the AWS Secrets Manger secret(s) to the seven secrets used by the OTIV deployment. These properties are:

global.otivSecretProviderClass.dbObjectName
The ARN for the AWS Secrets Manager secret containing the Database Admin password.

global.otivSecretProviderClass.otdsObjectName
The ARN for the AWS Secrets Manager secret containing the OTDS Admin password.

global.otivSecretProviderClass.ivObjectName
The ARN for the secret containing the RabbitMQ password, and the OTIV OAuth client secrets.

global.otivSecretProviderClass.objectName
Default secret ARN; must be set if one of the above properties isn't set.

Setting the `dbObjectName`, `otdsObjectName`, or `ivObjectName` properties is not a requirement, but if any of these are not set, the `objectName` property needs to be set so that the associated secret can be retrieved from it. For instance, if all seven Intelligent Viewing secrets are stored in the same AWS Secrets Manager secret, then only the `global.otivSecretProviderClass.objectName` property needs to be set.

Each of the seven Intelligent Viewing secrets has a helm property that defines the key for that Intelligent Viewing secret within the AWS Secrets Manager secret. See the OTIV chart's top level `values.yaml` file for a list of these properties (they end with "Key") and their default values.

6.3.2 Certificates

The OTIV chart's `global.otivSecretProviderClassCerts` properties listed in this section enable importing of certificates stored in AWS Secrets Manager secrets to OTIV containers' trust stores. Note that if a parent chart already creates a single Kubernetes secret that has all trust certificates, this functionality does not need to be enabled. Instead, the `global.enableOtivCustomizedTruststore` property can be set to the name of this secret.

Also note that the certificate should not be stored as a value within the AWS Secrets Manager secret's JSON structure. For instance, contents of a PEM certificate should be pasted into the plain text editor when using AWS console.

global.otivSecretProviderClassCerts.enabled
Set to `true` to enable importing of certificates stored in AWS Secrets Manager to OTIV container trust stores.

global.otivSecretProviderClassCerts.saRole
Set to the ARN of the IAM role that allows access to the AWS secrets.

global.otivSecretProviderClassCerts.secrets
An array of the AWS ARNs of the certificate secrets to be added to the OTIV containers' trust stores. An example of how to set this within a yaml file is:

```
global:

  otivSecretProviderClassCerts:
    secrets:
      - name: "arn:aws:secretsmanager:us-east-1:13539440:secret:otds/cert-GHDS"
        certFileName: otds.pem
      - name: "arn:aws:secretsmanager:us-east-1:13539440:secret:db/cert-R3WLB"
        certFileName: db.pem
```

6.3.3 Updates to AWS Secrets

The Secrets Store CSI driver, which is the technology that transfers the AWS Secrets Manager secrets to a Kubernetes secret, has a *--enable-secret-rotation* feature that, if set, refreshes the Kubernetes secret values shortly after an update is made to the secrets in AWS. However, the OTIV pods must still be restarted after the secret has been updated.

In the case that the *--enable-secret-rotation* feature is not enabled within the Secrets Store CSI driver (after an AWS secret has been updated), do the following:

1. Stop the OTIV pods
2. Delete the Kubernetes secret `otiv-spc-secrets`.
3. Restart the OTIV pods.

Part 2
Administration

Chapter 7

Monitoring considerations

The following endpoints are available to provide version information (an example of `<ingressDomainName>` would be `content.company.com`):

Service	URL
Publication	`https://otiv-publication-<namespace>.<ingressDomainName>/publication/api/v1/version`
Viewer	`https://otiv-viewer-<namespace>.<ingressDomainName>/viewer/api/v1/version`
Highlight	`https://otiv-highlight-<namespace>.<ingressDomainName>/search/api/v1/version`
Markup	`https://otiv-markup-<namespace>.<ingressDomainName>/markup/api/v1/version`
Asset	`https://otiv-asset-<namespace>.<ingressDomainName>/asset/api/v1/version`

In addition to the version endpoints, the API for the IV services provide health endpoints. These are accessed by replacing the `version` endpoint with either `health/ready` or `health/live`. For example: `https://otiv-viewer-<namespace>.<ingressDomainName>/viewer/api/v1/health/ready`

Each of these services have the following YAML properties defined, which can be adjusted as necessary:

```
livenessProbe:
  enabled: true
  initialDelaySeconds: <delay>
  timeoutSeconds: <timeout>
  periodSeconds: <period>
  failureThreshold: <failureCount>

readinessProbe:
  enabled: true
  initialDelaySeconds: <delay>
  timeoutSeconds: <timeout>
  periodSeconds: <period>
  failureThreshold: <failureCount>
```

If you need to restart an OTIV service, you can issue the kubectl rollout restart command:

```
kubectl rollout restart deployment otiv-config
```

Chapter 8

Licensing

8.1 Update a license

To update a license:

1. In the OTDS application, navigate to **License Keys**.
2. Click **Actions > Properties** and, on the left side, click **License Key**.
3. Update your license key using copy/paste, or click **Get License File** to browse for the new .lic file.
4. Click **Save**.

8.2 Assign a user to a license

The Intelligent Viewing license contains slots for two user-based features. The FULLTIME_USERS_BASIC license feature allows for the base level of functionality. This includes viewing documents, searching, and printing. The FULLTIME_USERS_REGULAR license feature includes the functionality provided for the FULLTIME_USERS_BASIC license feature, plus advanced functionality.

To assign a user to a license feature:

1. From the OTDS application, navigate to **Users & Groups**.
2. Click **Actions > Allocate to License** for a specific user.
3. In the **Allocate to License** dialog, select the **Viewing** license.
4. In the **Counter** field, select either INTELLIGENT_VIEWING.FULLTIME_USER_BASIC or INTELLIGENT_VIEWING.FULLTIME_USER_REGULAR.

Chapter 9

Upgrade the deployment

An update to the `imageTags.yaml` file, which contains the OTIV docker image tags to pull, is one scenario where an upgrade might be performed. Another situation is when an update has been made to one or more helm chart property values (for example, an update to `values.yaml` or to the sizing YAML files).

9.1 Review Persisted Volume Sizes

A helm upgrade will fail if the size of a persistent volume claim (PVC) associated with a stateful set (such as otds, otiv-amgp) changes. Note that the PVCs that are associated with deployments (the otiv-bas PVC) will not fail due to a change in the target PVC, provided that the storage class allows for volume upgrades. For a successful upgrade, you must update the existing sizes of the PVCs associated with the stateful sets to match the target sizes in the upgrade.

To update PVC sizes:

1. Check the size of the current PVCs by running the command:

   ```
   kubectl -n <namespace> get pvc
   ```

2. For otds and otiv-amqp (rabbitmq), compare the size to the target size. For example, if adding or changing a sizing file to the helm upgrade command (such as `-f medium.yaml`), review the storage size of the *`rabbitmq.persistence.size`* property and compare that to the size of the `data-otiv-amqp-0` PVC. If they differ, edit the PVC with the following command:

   ```
   kubectl edit pvc data-otiv-amqp-0
   ```

3. In your editor, update the value property *`resources.requests.storage`* to the target size (for example, `4Gi`), and save your changes.

   Note: You don't need to change the storage value under the `status` section.

4. To delete the stateful set without deleting the corresponding pods (to minimize downtime) run:

   ```
   kubectl delete sts --cascade=false otiv-amqp
   ```

9.2 Execute the helm upgrade

Note: Beginning with OTIV version 22.2, OTDS data is stored in a database. When migrating OTDS records from an earlier deployment to 22.2 or later, you must run a migration helm chart. Follow the instructions at the top of the `otdsMigrate.yaml` file, included in the OTIV helm chart artifact zip, to execute the data migration.

- If upgrading a development or test deployment from version 22.1 or earlier that uses the PostgreSQL container, execute the following commands prior to running the helm upgrade:

```
kubectl -n <namespace> delete sts pg-otiv
kubectl -n <namespace> delete secret pg-otiv
```

  As noted earlier, the use of PostgreSQL containers aren't supported in production deployments. Instead, a PostgreSQL instance should be set up outside of Kubernetes. For development and test purposes, a PostgreSQL container is provided.

- The PostgreSQL data dictionary has changed in versions 12 (introduced in IV 23.1) and 15 (introduced in IV 24.1). To perform a successful upgrade to an IV version across these boundaries, you must do ONE of the following:

  – Upgrade the data dictionary on the `data-pg-otiv-0` PersistentVolumeClaim (PVC). Required only if you need to keep your dev or test data. Refer to online search resources for instructions on upgrading PostgreSQL versions.

  – Maintain the earlier PostgreSQL version. For example, in the case of updating from 23.4 to 24.1, append the following arguments to the helm upgrade command:

```
--set postgresql.image.tag=14.5.0
```

  – Delete the `data-pg-otiv-0` PVC prior to upgrade. This can be accomplished with the following commands:

```
kubectl -n <namespace> scale --replicas=0 sts pg-otiv
# wait a minute until the pod terminates
kubectl -n <namespace> delete pvc data-pg-otiv-0
```

- Executing the following command will update affected OTIV Kubernetes resources:

```
helm upgrade -n <namespace> -i otiv ./otiv-25.4.0.tgz -f otiv/values.yaml -f otiv/
sizings/small.yaml
```

Chapter 10

Tips and troubleshooting

10.1 Pod status troubleshooting

If a problematic pod installation status persists (see "Review the deployment installation" on page 25), refer to the following table for troubleshooting assistance.

If all pods are down, the status of the required pods (such as OTDS, amqp, and Postgres) should be checked first.

Status	Potential causes	Troubleshooting
`ErrImagePull/ ImagePullBackOff` indicates that the worker nodes' docker daemon is unable to pull the docker image.	Path to docker image(s) is incorrect.	Run command `kubectl -n <namespace> describe pod <pod-name>` and check the bottom of the output for events entry `->Failed to pull image..` If no reason is found, adjust `global.imageSource` or `global.imageSourcePublic` properties.
	Kubernetes worker nodes have a network rule restricting access to external sites.	If a nodes external network access is restricted, ask the cluster administrator to update the nodes' docker daemons to use proxy.
	Credentials to access registry where IV images are stored are missing or incorrect.	If a credentials issue is present when retrieving docker images from `registry.opentext.com`, verify that a secret with your My Support credentials exists within the namespace and is referenced by `global.imgePullSecret`. For instructions for establishing a docker secret, see "Other requirements" on page 16. Note that the value of the `-docker-username` should be your My Support email.
`CrashLoopBackOff` indicates that the pod is crashing or terminating.	Container running out of disk space.	Capture logs through `kubectl -n <namespace> logs <pod-name>`.

Status	Potential causes	Troubleshooting
`Pending` indicates that the pod is stuck waiting for a resource to become available.	Not enough CPI or memory resources available in the cluster to schedule the pod. A Persistent Volume that the pods depend on can't be provisioned.	Check the pod's events with the command: `kubectl -n <namespace> describe pod <podname>` If there is a resource issue, you should see an event message that states either *Insufficient cpu* or *Insufficient memory*. To see the capacity of the cluster's nodes, execute the command: `kubectl describe node | grep -A5 "Allocated"` If the publication, publisher, and asset pods are all in the `Pending` state, a likely cause is that the PVC that they depend upon, `obiv-bas`, has not been provisioned. Check the PVC status using: `kubectl -n <namespace> get pvc` If the `obiv-bas` PVC status is in `Pending` state, it is likely that it couldn't be provisioned because the storage class provisioner specified by `global.storageClassNameNFS` doesn't provide ReadWriteMany (RWX) access. For information on ReadWriteMany storage classes, click here (https://medium.com/asl19-developers/create-readwritemany-persistentvolumeclaims-on-your-kubernetes-cluster-3a8db51f98e3).

10.2 Publishing requests unresponsive

The Intelligent Viewing transformation services (asset, config, publication, and publisher) form a hazelcast cluster, which is established for communication between these services. If the communication threads get blocked, Intelligent Viewing can be unresponsive to publishing requests. In this case, one troubleshooting tactic is to shutdown the Intelligent Viewing transformation pods and restart them. The script `otiv/scripts/restartTransformationPods.sh` is provided to facilitate this. This script takes a `<namespace>` argument that indicates the namespace where this operation is to be performed. There is also a corresponding PowerShell script (`otiv/scripts/restartTransformationPods.ps1`) provided for Windows.

10.3 Endpoint connection troubleshooting

To see the URL hosts of publicly accessed Intelligent Viewing endpoints, use the command:

```
kubectl -n <namespace> get ingress
```

You should see output similar to what is shown in "Add DNS records for EKS deployments" on page 25. If you are unable to connect to any of these endpoints, potential causes include:

- DNS record is invalid or non-existent.
- Mismatch in hostname spelling.
- Misconfiguration of nginx ingress controller.
- SSL Certificate issue.

To troubleshoot a connection, from a browser, attempt to launch the URL for the publication service: `https://otiv-publication-<namespace>.<ingressDomainName>/publication/api/v1/version`

- If a return message states that the site can't be reached or found, ensure that DNS records exist for the site. For example, using the table shown in "Add DNS records for EKS deployments" on page 25, it might be necessary to create a DNS A record associating the hostname (for example `*.sample.org`) to the IP address of the ingress controller, such as `10.100.200.41`.

  Verify that the host entered in the URL that you are trying to access matches the ingress host and the DNS record. Misspellings can cause mismatch errors.

- If a *502 Bad Gateway* message persists for several minutes after deployment, check the nginx logs to ensure that it's configured correctly:

  ```
  kubectl -n ingress-nginx logs <ingressControllerPodName>
  ```

- If a *503 Service Temporarily Unavailable* message displays, check the status on the target pod (`otiv-publication-..`) to make sure it's up and running.
- If an https error is reported, make sure that a wildcard SSL Kubernetes secret has been established, and that the value of the `global.ingressSSLSecret` property is set to this secret. See "Other requirements" on page 16.

10.4 File viewing troubleshooting

If a file is not displaying in the viewer, here are a few things you can try:

- Capture the logs for the otiv services by running the command:

  ```
  otiv/scripts/logs.sh <namespace>
  ```

  for Linux

  or

  ```
  otiv/scripts/logs.ps1 <namespace>
  ```

  for Windows

  Note: If the execution policy for your Windows user or machine is not set to `Unrestricted`, the command to run is: `powershell.exe -executionpolicy bypass -file .\logs.ps1 {Namespace}`

  The logs are written beneath the directory `./output/{namespace}/{mm-dd}`.

- The logging level for the service can be set. In `values.yaml`, see the `loglevel` property associated with each service. Changing the `loglevel` requires running a helm upgrade for the change to take affect. See "Updating the publisher log level" on page 48.

10.5 Updating the publisher log level

The following alternative methods are available for updating the Intelligent Viewing publisher containers log levels. The available log levels include : ERROR, WARNING, INFO, DEBUG, and TRACE.

- Update the log level in the running containers.
- Edit the publisher configmap.
- Run the helm upgrade command.

To update the log level in the running containers:

1. To update the publisher log level in the running publisher containers, the script `otiv/scripts/setPublisherLogLevel.sh`, located within the otiv helm chart (starting in 24.2), can be executed.
2. To extract the script, execute:

   ```
   tar xvfz otiv-25.4.0.tgz otiv/scripts/setPublisherLogLevel.sh
   ```

   The script's contents describe the publisher modules that can have their loglevel set.
3. As stated in the script's instructions, edit the script contents to set desired logging levels, then execute the script:

   ```
   otiv/scripts/setPublisherLogLevel.sh [ namespace ]
   ```

Usage note: The `[ namespace ]` argument is optional. If not provided, the operation is executed against the namespace for the current kubectl context.

The publisher containers are not restarted as a result of invoking this script. The updated log levels are in affect until the containers are restarted.

To edit the publisher configmap:

1. This method requires a relaunch of the publisher pods. Run the following command to scale down the publisher pods:

```
kubectl scale --replicas=0 deployment otiv-publisher
kubectl edit configmap otiv-publisher-configmap
```

2. Add the following lines to configmap and save the file:

```
  LOG_LEVEL_CONFIG: DEBUG
  LOG_LEVEL_CONVERSION: DEBUG
  LOG_LEVEL_METRICS: DEBUG
  LOG_LEVEL_NETTY: DEBUG
  LOG_LEVEL_RETRIEVAL: DEBUG
  LOG_LEVEL_SPI: DEBUG
  LOG_LEVEL_SPI_MKONDO: DEBUG
  LOG_LEVEL_SEQUENCING: DEBUG
  LOG_LEVEL_SYSTEM: DEBUG
  LOG_LEVEL_UTIL: DEBUG
  LOG_LEVEL_VERTX: DEBUG
```

3. To scale up the publisher pods, run the following command:

```
kubectl scale --replicas=1 deployment otiv-publisher
```

To run the helm upgrade command:

1. This method will relaunch the publisher containers. The publisher log level properties are listed with the file: `otiv/charts/publisher/values.yaml`. If `otiv` is a subchart of a parent helm chart, then the previous path will be underneath the parent's `charts` directory.

2. To update the logging for publisher, set the desired publisher log properties for the helm upgrade command. For example, the following arguments can be added to the helm upgrade command:

```
--set otiv.publisher.loglevel.config=DEBUG \
--set otiv.publisher.loglevel.conversion=DEBUG \
--set otiv.publisher.loglevel.metrics=DEBUG \
--set otiv.publisher.loglevel.netty=DEBUG \
--set otiv.publisher.loglevel.retrieval=DEBUG \
--set otiv.publisher.loglevel.sequencing=DEBUG \
--set otiv.publisher.loglevel.spi=DEBUG \
--set otiv.publisher.loglevel.spiMkondo=DEBUG \
--set otiv.publisher.loglevel.system=DEBUG \
--set otiv.publisher.loglevel.util=DEBUG \
--set otiv.publisher.loglevel.vertx=DEBUG
```

Note: The above properties assume that otiv is a subchart of another chart. If the otiv chart is being executed standalone, then the `otiv.` prefix can be removed from the beginning of the above properties.

10.6 Useful Kubernetes commands

The following table of commands can be helpful for general troubleshooting purposes.

Description	Command
To display a report of Kubernetes events across a namespace (all pods), sorted by timestamp:	`kubectl -n <namespace> get events --sort-by=.metadata.creationTimestamp`
To dump the YAML associated with a deployed helm chart:	`helm -n <namespace> get all`
To display the YAML for a Kubernetes resource type:	Append `-o yaml` to `kubectl get` (for example `kubectl -n <namespace> get pod otds-0 -o yaml`)
To report on the environmental variables in a pod:	`kubectl -n <namespace> exec <pod> -- env`
To access an internal service URL:	`kubectl -n <namespace> exec <publisherPod> -- curl http://otds/otdsws/rest/systemconfig/hello`
To ssh into a pod:	`kubectl -n <namespace> -it <podName> -- bash`
To restart an Intelligent Viewing pod:	`kubectl -n <namespace> rollout restart deployment <deploymentName>` Change `deployment` to `sts` for statefulsets (otds, otiv-amqp).
To see a list of deployments:	`kubectl -n <namespace> get deployments`