CUE Content Store

Installation Guide

7.15.12-3

Table of Contents

1 Introduction

This **Installation Guide** is intended to be read by the person responsible for installing the CUE Content Store on one or more hosts. It should be read together with the Server Administration Guide, which contains descriptions of the periodic administration tasks a system administrator needs to carry out once the Content Store is installed and in operation, since there is considerable overlap between installation and maintenance of the Content Store.

Both this manual and the **Server Administration Guide** make the following assumptions about the CUE installation and you, the reader:

- The Content Store is to be installed on one or more host computers running an operating system listed in section 1.1.2.
- The supporting software components (database, web server, application server and so on) listed in section 1.1.2 are either already installed on these computers, or available for installation.
- You are a suitably qualified system administrator with a working knowledge of both the operating system on which the Content Store is to be installed and of the components in the supporting software stack.

1.1 Installation Components

A working CUE Content Store installation depends on many components. Some of these components are delivered by Stibo DX as part of the product, others are third-party products on which the Content Store depends.

1.1.1 Escenic Components

The components supplied by Stibo DX are:

CUE Content Store
The Content Store itself, a content management system.

escenic-admin
A server administration web application.

Escenic Web Studio
A publication administration web application.

Publications
The installation includes a few small demo publications for test, demonstration and instructional purposes.

1.1.2 Third-Party Components

The Content Store depends on the following third-party components:

Java Development Kit (JDK)
This provides the Java runtime and development environment required by J2EE application servers.

Database
Used to store publication content, structure and so on.

J2EE Java application server
The Content Store is a JEE web application, and therefore runs inside a JEE application server. `escenic-admin`, Web Studio and all CUE publications are also JEE applications and therefore also run within a JEE application server.

JDBC (Java DataBase Connectivity) driver
This is required to connect the Content Store to the database.

Apache Solr
This is a Java-based search engine. It is used to provide search functionality both in web publications and in CUE.

Caching Server (optional)
A caching server such as Squid or Varnish can greatly improve the performance of heavily-loaded sites by caching frequently requested pages and sending the cached copies on request unless the original pages have been updated.

Load Balancing Server (optional)
If your site has multiple web hosts, then you may also want to install a load-balancing server to distribute requests evenly between them.

memcached (optional)
If your site has multiple web hosts, then you can improve caching efficiency by using this open source distributed cache manager to provide a shared cache for all your hosts.

In principle, the Content Store can work with many different third-party components. In section 2.1, however, you will find the list of officially supported software. The current version of the Content Store has been tested with these components and is known to work satisfactorily.

1.2 Installation Types

The Content Store can be installed in a number of different ways to satisfy different needs. Common installation types include:

Single computer
Everything (database, application server, Content Store) is installed on a single computer. This is usually the case for demonstration and developer installations.

Multiple application servers, single database
The application server and Content Store is installed on multiple computers, but all share a single database server. This kind of installation is common in large development environments.

Single web server, single application server and Content Store, single database
This is a common production installation structure for small sites.

Multiple web servers, multiple application servers/Content Stores, database rack
This is a common production installation structure for large sites, and is easily scalable.

Other configurations are, of course, possible.

In large organizations the Content Store often needs to be installed in several different configurations in order to meet the requirements of different environments:

Development
In a development environment there are typically several updates of several files several times a day. Depending on the number of developers working on a system and the potential for conflicting changes, developers may either:

- Share a single development installation, or
- Each use a personal Content Store installation but share a single database.

Test
A test environment is a complete Content Store installation on which 'current stable versions' can be tested. Code updates in this environment are typically carried out at longer intervals and in 'batches' rather than on the file level. In some organizations, the test environment may be merged with the staging environment.

Staging
A staging environment should be as similar to the production environment as practically possible: it should have the same software, the same configuration, and if possible the same hardware, firewall and network setup. This is the last stage before actual production, and is intended to catch "works on the test server but not in production" problems.

Production
A production environment is heavily optimized for high workloads and security.

1.3 Conventions Used in This Manual

This section describes various conventions used in this manual.

1.3.1 Identifying Host Machines

The Content Store and the software it depends on may need to be installed on one or several host machines depending on the type of installation required (see section 1.2). In order to unambiguously identify the machines on which various installation actions must be carried out, the following special host names are used throughout this manual:

database-host
The machine used to host the database server.

engine-host
The machine(s) used to host application servers and Content Store instances.

editorial-host
engine-host(s) that are used solely for (internal) editorial purposes. All your CUE clients communicate with these hosts.

presentation-host
engine-host(s) that are used solely for (public) presentation purposes. Readers' browsers communicate with these hosts.

web-host
The machine(s) used to host web servers (if needed).

share-host
The machine used to host the NFS server (if needed).

These host names always appear in a **bold typeface**. If you are installing everything on one host you can, of course, ignore them: you will be doing everything on the same machine. If you are creating a larger multi-host installation, then they should help ensure that you do things in the right places.

1.3.2 Shell Commands

All shell command examples given in the manual are tested on Debian Linux servers: they may need minor modifications to be used on other Linux or UNIX platforms, and it is assumed that you are able to make the necessary "conversions" to your own platform. Some of the commands should be executed as the owner of the Content Store installation. This is signalled by use of the **$** command prompt. For example:

```
$ ls
```

Other commands must be executed as **root**. This is signalled by the use of the **#** command prompt:

```
# /etc/init.d/slapd restart
```

Many of the code and command line examples in the manual contain placeholders: words printed in an *italic* typeface, which you are expected to replace with an appropriate value. For example:

```
$ cat filename
```

The accompanying text usually specifies what kind of value any placeholder(s) represent: *filename* in this case is the name of a file you want to see the contents of.

1.3.3 Standard File Hierarchy

All file paths and URLs shown in the manual are based on the following standard folder structure.

Standard location	Component
/usr/share/escenic	CUE
/usr/share/escenic/escenic-content-engine-7.15.12-3	CUE Content Store
/usr/share/escenic/ece-scripts	CUE scripts
/etc/escenic	CUE configuration
/etc/escenic/engine	CUE Content Store configuration
/opt/java/jdk	Java
/opt/tomcat-engine1	Tomcat

If your system is organized differently, then adjust the paths you use accordingly.

2 Product Requirements

The CUE Content Store is written in Java, runs on an application server, and uses a database to store content and user data plus a search engine to provide search functionality. One of each of these supporting products must be installed before the Content Store can be installed. The specific supporting products with which the Content Store is known to work are listed in section 2.1. Note, however, that these products have their own hardware and software requirements. Please make sure that your environment meets the requirements stipulated by the relevant product suppliers.

Some additional, more general requirements and recommendations are listed in section 2.2.

2.1 Required Supporting Software

The Content Store requires a combination of the following software components:

Operating System
One of the following:

- Ubuntu Server 18.04 LTS (Long Term Support)
- Red Hat Enterprise Linux 7.4

Application Server
Apache Tomcat 9.0.x

Database Server
MariaDB 10.x

Java Development Kit
JDK 11 - either OpenJDK 11 or Oracle Java SE Development Kit 11

Search Engine
Solr 8.7.0

2.2 Other Requirements

2.2.1 SSH/FTP Server

The Content Store does not need an SSH or FTP Server. Setting up an SSH or FTP server is, however, highly recommended. An SSH/FTP Server can be used to upload new and updated files to the server. If it is not possible to install an SSH or FTP server on the **engine-host** (for security reasons, perhaps), then some other way of uploading files must be provided.

3 Installation Procedure

This chapter contains step-by-step instructions for installing the Content Store on a single host computer or on a cluster of several host computers. The instructions use the host names listed in section 1.3.1 to indicate where you should carry out various steps. If some of your hosts are "multi-purpose" (if for example, your database server is installed on the same host as your editorial Content Store installation), then carry out all appropriate steps on that machine (for example, all **database-host** steps and all **editorial-host** steps).

Some of the steps are not required for single-host deployment. These steps are clearly marked.

A note about version code names

The CUE Content Store and all related applications (plug-ins, the CUE editor, CUE Print, CUE Front and so on) are released on a synchronized schedule where all product versions in a given release are known to work well together. Only these approved version combinations are supported. Each set of compatible product versions is identified by a code name, and during installation you can use this code name instead of the individual product's version number, thereby simplifying the installation process.

In the case of the Content Store and other Linux applications installed on Ubuntu using **`apt-get`**, the code name is actually the name of a repository containing compatible versions of all products. This means that in order to ensure version compatibility, all you need to do is add the name of the required repository to your **`/etc/apt/sources.list.d/escenic.list`** file. Once you have done this you do not need to specify any version numbers when installing individual packages - **`apt-get`** will just install the latest maintenance release from that repository.

> Note that code names cannot be used in this way on Red Hat installations, where the application packages to be installed must still be identified by their version numbers.

The code name for Content Store 7.12 is **`lithium`**.

3.1 Install Java Development Kit (JDK)

On your **engine-host**(s) while logged in as **`root`**, install version 11 of the OpenJDK Java development kit as follows:

```
# apt-get install openjdk-11-jdk
```

> If you wish to install the Oracle SE JDK instead of OpenJDK, follow the installation instructions supplied on the Oracle SE JDK web site.

Verify that Java is correctly installed by entering:

```
# java -version
```

3.2 Install Various Utilities

Make sure the following utilities are available on all your hosts:

- **unzip**
- **telnet**

On Debian-based Linux machines, you can easily install these by entering the following commands as **root**:

```
# apt-get install unzip
# apt-get install telnet
```

You may also find it useful to install an SSH server on all your hosts so that you can easily switch sessions and copy files between them. On Debian-based Linux machines, you can do this as follows:

```
# apt-get install openssh-server
```

3.3 Create CUE Users

All the Content Store components should run under the same user, and this user should have the same UID on all hosts. So create a user (preferably called **escenic**) on each host machine as follows:

```
# adduser escenic
```

and make sure they all have the same UID. If you are installing on clean machines, and **escenic** is the first user you create, then they will all have the same UID by default.

On your **engine-host**(s), make sure that Java is in the path of the **escenic** users:

```
# su - escenic
$ echo 'export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64' >> .bashrc
$ echo 'export PATH=$JAVA_HOME/bin:$PATH' >> .bashrc
$ source ~/.bashrc
```

> If you have installed the Oracle JVM then you will need to replace **/usr/lib/jvm/java-11-openjdk-amd64** in the above commands with the correct Oracle installation path.

3.4 Install CUE Packages

To install the required CUE packages on your **engine-host**(s), log in as **root** on each machine and perform the following steps:

1. If necessary, add the Escenic repository signing key to your machine's keyring:

   ```
   # curl --silent https://apt.escenic.com/repo.key | apt-key add -
   ```

2. Add the current version repository name to your list of sources.

   ```
   # echo "deb https://user:password@apt.escenic.com lithium main non-free" >> /etc/
   apt/sources.list.d/escenic.list
   ```

   where *user* and *password* are your CUE download credentials. If you do not have any download credentials, please contact CUE support.

3. Update your package lists and install the Content Store and the CUE scripts:

   ```
   # apt-get update
   # apt-get install escenic-content-engine
   ```

```
# apt-get install escenic-content-engine-scripts
# apt-get install escenic-content-engine-updater
# apt-get install plug-in-packages
```

where *plug-in-packages* is a list of any CUE plug-in packages that you want to install together with the Content Store.exit

The commands you need to use to install these packages on RedHat systems are:

```
# rpm -Uvh https://user:password:yum.escenic.com/rpm/escenic-content-
engine-7.12-7.15.12-3.x86_64.rpm
# rpm -Uvh https://user:password:yum.escenic.com/rpm/escenic-content-engine-
scripts-7.12-7.15.12-3.x86_64.rpm
# rpm -Uvh https://user:password:yum.escenic.com/rpm/escenic-content-engine-
updater-7.12-7.15.12-3.x86_64.rpm
```

where *version* is the correct version number of the package.

3.5 Install Database

The following instructions describe how to install and set up MariaDB for use by the Content Store.

On your **database-host**, while logged in as **`root`**:

1. Install the MariaDB server and client packages. For example, on a Debian-based Linux distribution:

   ```
   # apt-get install mariadb-server mariadb-client
   ```

2. Log in to the system and create a database for the Content Store:

   ```
   # mysql
   mysql> create database db-name character set utf8 collate utf8_general_ci;
   Query OK, 1 row affected (0.00 sec)

   mysql> grant all on db-name.* to user@'%' identified by 'password';
   Query OK, 0 rows affected (0.00 sec)

   mysql> grant all on db-name.* to user@'localhost' identified by 'password';
   Query OK, 0 rows affected (0.00 sec)
   mysql> exit
   ```

   Replace *db-name*, *user* and *password* in the above commands with a database name, user name and password of your choice.

3. Change user to **`escenic`** and run the Content Store's database scripts:

   ```
   # su - escenic
   $ cd /usr/share/escenic/escenic-content-engine-7.12/database/mysql/
   $ for el in tables.sql indexes.sql constants.sql constraints.sql; do \
       mysql -u user -ppassword db-name < $el
       done;
   ```

   Replace *db-name*, *user* and *password* in the above commands with the names you chose in step 2. Your *password* must **immediately follow** the **`-p`** switch, with no intervening space (as shown above).

4. On some platforms, external access to the MariaDB server is disabled by default. To enable external access, you need to bind the **`mysql`** process to the **database-host**'s IP address. To do this, you need to open **`/etc/mysql/mariadb.conf.d/50-server.cnf`** for editing (as **`root`** again) and set the **`bind-address`** parameter:

   ```
   bind-address = database-host-ip-address
   ```

You should then verify that the MariaDB server is running and accessible by trying to connect to port 3306 from each of your other hosts using **`telnet`**:

```
$ telnet database-host 3306
```

where *database-host* is the host name or IP address of the **database-host**. If a connection is opened, then the database server is running and accessible.

On a single-host installation, you can check that MariaDB is running by entering the following command as **`root`**:

```
# mysqladmin status
```

You should get a response something like this:

```
Uptime: 605 Threads: 1 Questions: 615 Slow queries: 0 Opens: 842 Flush tables: 1 Open
 tables: 40 Queries per second avg: 1.16
```

If the server is not running then you will see an error reporting that it was not possible to connect to the server.

3.6 Install Application Server

The following instructions describe how to install and set up the Apache Tomcat application server for use by the Content Store.

On your **engine-host**(s), while logged in as **`root`**:

1. Download a Tomcat package from http://tomcat.apache.org/ to a temporary location. Make sure that you download the version specified in section 2.1.
2. Unpack the Tomcat package to **`/opt`**:

   ```
   # cd /opt
   # tar -zxvf /tmp/apache-tomcat-version-number.tar.gz
   ```

   where *version-number* is the version number of the package you have downloaded.
3. Make **`/opt/apache-tomcat-`***version-number* readable and executable by all users:

   ```
   chmod -R 755 /opt/apache-tomcat-version-number
   ```
4. Create a symbolic link from **`/opt/tomcat`** to the Tomcat folder, so that it will be easier to upgrade to new versions when necessary:

   ```
   # ln -s /opt/apache-tomcat-version-number /opt/tomcat
   ```
5. Configure two Tomcat instances: one to hold the main Content Store web applications (see section 3.6.1), and one to hold search-related web applications (see section 3.6.2).

3.6.1 Configure Main Tomcat Instance

On your **engine-host**(s), while logged in as **`root`**:

1. Create a Tomcat instance folder called **`tomcat-engine1`**:

   ```
   # mkdir -p /opt/tomcat-engine1
   ```

2. Copy the Tomcat **`conf`** folder and its contents to the new instance folder you have created:

   ```
   # cp -r /opt/tomcat/conf /opt/tomcat-engine1/
   ```

3. Create the other standard folders required in a Tomcat instance folder:

   ```
   # mkdir -p /opt/tomcat-engine1/{lib,logs,temp,webapps,work}
   ```

4. Open the Tomcat configuration file **`/opt/tomcat-engine1/conf/catalina.properties`** for editing and add the following characters:

   ```
   ,"${catalina.base}/escenic/lib/*.jar"
   ```

   to the end of the **`common.loader`** property setting.

5. Create the **`/opt/tomcat-engine1/escenic/lib`** folder you added to the **`common.loaderpath`** in step 4:

   ```
   # mkdir -p /opt/tomcat-engine1/escenic/lib
   ```

6. Download the latest version of **Connector/J** (the MariaDB JDBC driver) from https://downloads.mariadb.org/connector-java/ to a temporary location (**`/tmp`** for example).

7. Create an **`/opt/tomcat-engine1/lib/`** folder and copy the downloaded driver into it:

   ```
   # mkdir -p /opt/tomcat-engine1/lib
   # cp /tmp/mariadb-java-client-latest-version.jar /opt/tomcat-engine1/lib/
   ```

8. Open **`/opt/tomcat-engine1/conf/server.xml`** for editing and replace the entire contents with the following:

   ```
   <?xml version="1.0" encoding="utf-8"?>
   <Server port="8005" shutdown="SHUTDOWN">
     <Listener
       className="org.apache.catalina.core.AprLifecycleListener"
       SSLEngine="on"/>
     <Listener
       className="org.apache.catalina.core.JreMemoryLeakPreventionListener"/>
     <Listener
       className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener"/>
     <GlobalNamingResources>
       <Resource
         name="UserDatabase"
         auth="Container"
         type="org.apache.catalina.UserDatabase"
         description="User database that can be updated and saved"
         factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
         pathname="conf/tomcat-users.xml"/>
       <Resource
         name="jdbc/ECE_DS"
         auth="Container"
         type="javax.sql.DataSource"
         factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
         maxActive="500"
         maxIdle="8"
         maxWait="2000"
         initialSize="20"
         username="user"
   ```

```
      password="password"
      driverClassName="org.mariadb.jdbc.Driver"
      url="jdbc:mysql://database-host-and-port/db-name?
autoReconnect=true&amp;useUnicode=true&amp;characterEncoding=UTF-8&amp;characterSetResults=UTF-8
      testOnBorrow="false"
      testOnReturn="false"
      timeBetweenEvictionRunsMillis="60000"
      numTestsPerEvictionRun="5"
      minEvictableIdleTimeMillis="30000"
      testWhileIdle="true"
      validationQuery="select now()"/>
  </GlobalNamingResources>
  <Service name="Catalina">
    <Connector
      port="8080"
      protocol="org.apache.coyote.http11.Http11NioProtocol"
      connectionTimeout="20000"
      URIEncoding="UTF-8"
      compression="off"
      redirectPort="8443"/>
    <Connector port="8443"
      protocol="org.apache.coyote.http11.Http11NioProtocol"
      connectionTimeout="20000"
      URIEncoding="UTF-8"
      proxyPort="443"
      scheme="https"/>
    <Connector port="8083"
      protocol="org.apache.coyote.http11.Http11NioProtocol"
      connectionTimeout="20000"
      URIEncoding="UTF-8"
      compression="off"
      redirectPort="8443"
      proxyPort="443"
      scheme="https"/>
    <Engine name="Catalina"
      defaultHost="localhost"
      jvmRoute="jvm1">
      <Valve
        className="org.apache.catalina.valves.AccessLogValve"
        prefix="access"
        suffix=".log"
        pattern="common"/>
      <Realm
        className="org.apache.catalina.realm.UserDatabaseRealm"
        digest="md5"
        resourceName="UserDatabase"/>
      <Host
        name="localhost"
        appBase="webapps"
        unpackWARs="true"
        autoDeploy="true"
        xmlValidation="false"
        xmlNamespaceAware="false">
      </Host>
    </Engine>
  </Service>
```

```
</Server>
```

Replace *database-host-and-port*, *db-name*, *user* and *password* in the above definitions with the names you have defined earlier (see section 3.5).

> If you copy and paste the above code, make sure to remove any line breaks in the **`url`** values (caused by page width restrictions).

9. Open **`/opt/tomcat-engine1/conf/context.xml`** for editing and replace the entire contents with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
  <WatchedResource>WEB-INF/web.xml</WatchedResource>
  <WatchedResource>WEB-INF/tomcat-web.xml</WatchedResource>
  <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>
  <ResourceLink global="jdbc/ECE_DS" name="jdbc/ECE_DS"
 type="javax.sql.DataSource"/>
  <Environment name="escenic/solr-base-uri"
    value="http://localhost:8983/solr"
    type="java.lang.String"
    override="false"/>
  <JarScanner>
    <JarScanFilter defaultPluggabilityScan="false"/>
  </JarScanner>
</Context>
```

10. Change the owner of the instance folder to **`escenic`**:

```
# chown -R escenic:escenic /opt/tomcat-engine1
```

3.6.2 Configure Search Tomcat Instance

On your **engine-host**(s), while logged in as **`root`**:

1. Create a second Tomcat instance called **`tomcat-search1`** by copying the entire **`tomcat-engine1`** folder tree:

```
# cp -r /opt/tomcat-engine1 /opt/tomcat-search1
```

2. Open **`/opt/tomcat-search1/conf/server.xml`** for editing and replace the entire contents with the following:

```
<?xml version='1.0' encoding='utf-8'?>
<Server port="8150" shutdown="SHUTDOWN">
  <Listener
    className="org.apache.catalina.core.AprLifecycleListener"
    SSLEngine="on" />
  <Listener
    className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
  <Listener
    className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />

  <GlobalNamingResources>
    <Resource
      name="UserDatabase"
      auth="Container"
      type="org.apache.catalina.UserDatabase"
      description="User database that can be updated and saved"
      factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
      pathname="conf/tomcat-users.xml"
    />
```

```
  </GlobalNamingResources>

  <Service name="Catalina">
    <Connector
      port="8180"
      protocol="org.apache.coyote.http11.Http11NioProtocol"
      connectionTimeout="20000"
      URIEncoding="UTF-8"
      compression="off"
      redirectPort="8133"
    />
    <Connector
      port="8133"
      protocol="org.apache.coyote.http11.Http11NioProtocol"
      connectionTimeout="20000"
      URIEncoding="UTF-8"
      proxyPort="443"
      scheme="https"
    />
    <Engine
      name="Catalina"
      defaultHost="localhost"
      jvmRoute="jvm1">
      <Valve
        className="org.apache.catalina.valves.AccessLogValve"
        prefix="access"
        suffix=".log"
        pattern="common"/>
      <Realm
        className="org.apache.catalina.realm.UserDatabaseRealm"
        digest="md5"
        resourceName="UserDatabase"/>
      <Host
        name="localhost"
        appBase="webapps"
        unpackWARs="true"
        autoDeploy="true"
        xmlValidation="false"
        xmlNamespaceAware="false">
      </Host>
    </Engine>
  </Service>
</Server>
```

3. Open **`/opt/tomcat-search1/conf/context.xml`** for editing and replace the entire contents with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
  <WatchedResource>WEB-INF/web.xml</WatchedResource>
  <WatchedResource>WEB-INF/tomcat-web.xml</WatchedResource>
  <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>
  <Environment
    name="escenic/solr-base-uri"
    value="http://localhost:8983/solr"
    type="java.lang.String"
    override="false"/>
  <Environment
    name="escenic/indexer-webservice"
    value="http://indexer-web-service-host:8080/indexer-webservice/web-service-
name/"
```

```
    type="java.lang.String"
    override="false"/>
  <Environment
    name="escenic/index-update-uri"
    value="http://localhost:8983/solr/solr-core/update/"
    type="java.lang.String"
    override="false"/>
  <Environment
    name="escenic/head-tail-storage-file"
    value="/var/lib/escenic/head-tail.index"
    type="java.lang.String"
    override="false"/>
  <Environment
    name="escenic/failing-documents-storage-file"
    value="/var/lib/escenic/failures.index"
    type="java.lang.String"
    override="false"/>
</Context>
```

where:

- On an **editorial-host**, *indexer-web-service-host* is the host name or IP address of the **editorial-host** on which the Content Store's internal indexer web service is to run and *web-service-name* is **`index`**.
- On a **presentation-host**, *indexer-web-service-host* is the host name or IP address of the **presentation-host** on which the Content Store's external indexer web service is to run and *web-service-name* is **`presentation-index`**.
- The *solr-core* is usually either **`editorial`** or **`presentation`**.

If you are creating a single-host installation, then you can use the host name **`localhost`** and the indexer web service name **`index`**.

> Make sure that your **`escenic/solr-base-uri`** setting in **`/opt/tomcat-search1/conf/context.xml`** is as shown, and **does not** include a trailing slash (**`/`**). If it does, then search will not work.

4. Create a folder called **`/opt/tomcat-search1/conf/Catalina/localhost/`**:

   ```
   # mkdir -p /opt/tomcat-search1/conf/Catalina/localhost/
   ```

5. Create an empty file in the folder called **`indexer-webapp-`***engine-host-type***`.xml`** where *engine-host-type* is either **`editorial`** or **`presentation`**:

   ```
   # touch /opt/tomcat-search1/conf/Catalina/localhost/indexer-webapp-engine-host-type.xml
   ```

6. Open **`indexer-webapp-`***engine-host-type***`.xml`** in an editor and add the following contents:

   ```
   <?xml version="1.0"?>
   <Context docBase="$\{catalina.base}/webapps/indexer-webapp">
     <Environment
         name="escenic/solr-base-uri"
         value="http://indexer-web-service-host:8983/solr/solr-core"
         type="java.lang.String"
         override="true"/>
     <Environment
         name="escenic/indexer-webservice"
         value="http://indexer-web-service-host:8080/indexer-webservice/web-service-name/"
         type="java.lang.String"
         override="true"/>
   ```

```
  <Environment
      name="escenic/index-update-uri"
      value="http://indexer-web-service-host:8983/solr/solr-core/update/"
      type="java.lang.String"
      override="true"/>
  <Environment
      name="escenic/head-tail-storage-file"
      value="/var/lib/escenic/engine/head-tail.index"
      type="java.lang.String"
      override="true"/>
  <Environment
      name="escenic/failing-documents-storage-file"
      value="/var/lib/escenic/engine/failures.index"
      type="java.lang.String"
      override="true"/>
</Context>
```

where:

- On an **editorial-host**, *indexer-web-service-host* is the host name or IP address of the **editorial-host** on which the Content Store's internal indexer web service is to run and *web-service-name* is **`index`**.
- On a **presentation-host**, *indexer-web-service-host* is the host name or IP address of the **presentation-host** on which the Content Store's external indexer web service is to run and *web-service-name* is **`presentation-index`**.
- The *solr-core* is usually either **`editorial`** or **`presentation`**.

7. Change the owner of the instance folder to **`escenic`**:

```
# chown -R escenic:escenic /opt/tomcat-search1
```

3.7 Logging

On one of your **engine-host**s, while logged in as **`root`**:

1. Create the folder **`/etc/escenic/engine/common`**:

```
# mkdir -p /etc/escenic/engine/common
```

2. Create a logging configuration file called **`trace.properties`** in the new folder. This file should have the following content:

```
rootLogger.level=ERROR
  rootLogger.appenderRef.rolling.ref = ECELOG

  logger.escenic.name=com.escenic
  logger.escenic.level=ERROR
  logger.escenic.appenderRef.rolling.ref = ECELOG

  logger.neo.name=neo
  logger.neo.level=ERROR
  logger.neo.appenderRef.rolling.ref = ECELOG

  appender.ECELOG.name = ECELOG
  appender.ECELOG.type=RollingFile
  appender.ECELOG.filename=/var/log/escenic/engine/ece-messages.log
  appender.ECELOG.filePattern = /var/log/escenic/engine/ece-messages.%d{yyyy-MM-
dd}.log
```

```
appender.ECELOG.policies.type = Policies
appender.ECELOG.policies.time.type = TimeBasedTriggeringPolicy
appender.ECELOG.policies.time.interval = 1

appender.ECELOG.layout.type=PatternLayout
appender.ECELOG.layout.pattern=%d %5p [%t] %x (%c) %m%n
```

On all your **engine-host**s, while logged in as **`root`**:

1. Create an **`escenic`** log folder:

   ```
   # mkdir -p /var/log/escenic
   # chown -R escenic:escenic /var/log/escenic
   ```

2. Switch user to **`escenic`**:

   ```
   # su - escenic
   ```

3. Create the output folder specified in **`trace.properties`**:

   ```
   $ mkdir -p /var/log/escenic/engine
   ```

4. In order to make Tomcat use the **`trace.properties`** file, create a link to it from **`/opt/tomcat-engine1/lib`**:

   ```
   $ cd /opt/tomcat-engine1/lib
   $ ln -s /etc/escenic/engine/common/trace.properties
   ```

5. In order to make Tomcat output its log files to a standard Unix/Linux location, open **`/opt/tomcat-engine1/conf/logging.properties`** for editing and set **all** the log folder properties (they have names ending with **`FileHandler.directory`**) to **`/var/log/tomcat`**.

3.8 Install Solr

On your **engine-host**(s), while logged in as **`root`**:

1. Download a Solr package from a suitable Apache mirror site to a temporary location. Make sure that you download the version specified in section 2.1.
2. Unpack the Solr package to **`/opt`**:

   ```
   # cd /opt
   # unzip /tmp/solr-solr-version.zip
   ```

   where *solr-version* is the version number of the package you have downloaded.
3. Create a symlink **`/opt/solr`** for the new installation folder:

   ```
   # ln -s solr-solr-version solr
   ```

4. Create the configuration and start-up files required to manage the Solr service, as described in section 3.8.1.
5. Create a Solr **core** by copying the Solr configuration files supplied with the Content Store to a suitable location on the host, as described in section 3.8.2.

> In Solr terminology, a **`core`** is the set of configuration files that define a specific application of Solr – a schema, an index and all related configuration files. A typical CUE production installation makes use of two Solr cores – one for editorial purposes and one for presentation purposes. For further information about this, see Customizing the Index Schema..

3.8.1 Create Solr Startup Files

To create the necessary Solr startup files:

1. Create a Solr core folder and **`core.properties`** file as follows:

   ```
   #  mkdir -p /var/lib/escenic/solr/solr-core
   #  cat > /var/lib/escenic/solr/solr-core/core.properties <<EOF
   name=solr-core
   config=solrconfig.xml
   schema=schema.xml
   dataDir=data
   EOF
   ```

   where *solr-core* is the name of your Solr core. Typically you might choose to use either **`editorial`** or **`presentation`** as the name of your core depending on its purpose (that is, what kind of host you are installing it on).

2. Change ownership of the Solr folder and all its contents, so that it can be used by the **`escenic`** user:

   ```
   # chown -R escenic:escenic /var/lib/escenic/solr
   ```

3. Create a configuration file for the **`/opt/solr/bin/solr`** startup script. Do this by **moving** (not copying) the file as follows:

   ```
   # mv /opt/solr/bin/solr.in.sh /etc/default/
   ```

4. Append settings for **`SOLR_HOME`**, **`SOLR_LOGS_DIR`** and **`SOLR_PID_DIR`** to the moved script as shown below:

   ```
   # cat >> /etc/default/solr.in.sh <<EOF
   SOLR_HOME=/var/lib/escenic
   SOLR_LOGS_DIR=/var/log/escenic
   SOLR_PID_DIR=/var/run/escenic
   EOF
   ```

   This will ensure that Solr looks for the core in the right place (where you created it in step 1) and writes log files and runtime files to the same locations as corresponding Content Store files.

5. Make sure that all the folders you specified in step 4 actually exist and that the **`escenic`** user is either the owner or has read/write permissions.

6. Add an **`init.d`** script for starting Solr automatically on boot:

   ```
   # cp /opt/solr/bin/init.d/solr /etc/init.d/
   # sed -i "s#RUNAS=\"solr\"#RUNAS=\"escenic\"#" /etc/init.d/solr
   ```

7. Add the **`init.d`** script to the desired run levels. For Debian- and Ubuntu-based systems, you can do this as follows:

   ```
   # update-rc.d solr defaults 35
   ```

   For other systems, please consult the appropriate documentation.

3.8.2 Copy Solr Configuration

You now need to copy the following Solr configuration data:

- The main Solr configuration file supplied with Solr itself
- Solr core configuration files supplied with the Content Store.

The configuration data should be copied into a folder under **/etc/escenic** and then linked to from the **/var/lib/escenic/solr/** tree as described below.

For a single host installation:

1. Log in as **root** and create a **solr** folder under **/etc/escenic**, then copy the **solr.xml** file supplied with Solr to **/etc/escenic**:

   ```
   # mkdir -p /etc/escenic/solr
   # cp /opt/solr-solr-version/server/solr/solr.xml /etc/escenic
   ```

   where *solr-version* is the version number of your Solr installation.

2. Copy the configuration folder supplied with the Content Store:

   ```
   # cp -r /usr/share/escenic/escenic-content-engine-7.12/solr/conf /etc/escenic/
   solr/solr-core
   ```

   where *solr-core* is the name of your Solr core (most likely **editorial** on a single host installation).

3. Create a link to the copied file as follows:

   ```
   $ cd /var/lib/escenic/
   $ ln -s /etc/escenic/solr.xml
   ```

4. Log in as **escenic** and create a link to the copied folder as follows:

   ```
   $ cd /var/lib/escenic/solr/solr-core
   $ ln -s /etc/escenic/solr/solr-core conf
   ```

For a multiple host installation you will need to repeat these steps on each host, and in this case, *solr-core* is likely to vary according to the type of **engine host** – either **editorial** or **presentation**.

Note that the Solr configuration data supplied with the Content Store is intended for use on an editorial host and is not ideally suited to presentation hosts, so for production installations, it will need to be modified. For more about this, see Customizing the Index Schema..

3.9 Configure Content Store

The Content Store installation includes a default set of configuration files which you will find in **/etc/escenic/engine/common**. All the property settings in these files are commented out. You need to uncomment and set values in some of the files.

On each of your **engine-host**s, while logged in as **root**:

1. Open **/etc/escenic/engine/common/ServerConfig.properties** for editing, and make sure the following properties are present, uncommented and set correctly for your site:

   databaseProductName
   The name of the database you are using. Currently allowed values are:

   - **MariaDB**

   filePublicationRoot
   The path of the folder in which all binary files served by the Content Store are stored (multimedia files - video, audio, images, Word documents, PDF files and so on). You should set this to:

   ```
   filePublicationRoot=/var/lib/escenic/engine/
   ```

   The path you specify **must** include a trailing slash.

   webPublicationRoot
   The URL root you want to be used for all your publications. For example **http://***my-company***.com/**. The URLs of your publications are formed by appending the publication name to this string. This property is in general used only for development purposes. In production, it is often the case that each publication has its own root URL.

   customerId
   The name of your organization, or some other name that will clearly identify your installation to Stibo DX support staff, plus information about the type of this Content Store instance. This identifier is used for error and usage reporting. For production installations you **must** use the following format:

   ```
   customerId=prod@customer-id
   ```

   where *customer-id* is an identifier for your organization - your domain name for example:

   ```
   customerId=prod@mycompany.com
   ```

   You can optionally prefix the customer ID with a qualifier of some kind if you have multiple installations and separate reporting is considered useful. You might, for example, replace **prod@mycompany.com** with **prod@dailynews.mycompany.com** on some hosts and **prod@weeklymag.mycompany.com** on others.

   For non-production instances, you can replace the **prod** instance type identifier with another identifier of your choice, such as **test** or **staging**.

   Reporter components that are installed with the Content Store automatically send reports about the Stibo DX software running at your site. The reported information includes:

   - Which versions of the Content Store and its plug-ins are running at your site.
   - The number of active CUE users (reported once an hour)

   This information is very valuable to Stibo DX support engineers and helps with the diagnosis of reported errors and performance problems. It is also used in the calculation of support and maintenance charges. For this reason your are **required** to set this property on all Content Store installations and to configure your firewall so that reports can be delivered (see below).

   The reporter components sends their reports to Google Analytics. In order for this to work, your firewalls must allow outgoing traffic on port 80 to **http://www.google-analytics.com**.

2. Open **/etc/escenic/engine/common/neo/io/managers/ContentManager.properties** for editing, and make sure the following properties are uncommented and set correctly:

   dataConnector
   Must be set as follows:

   ```
   dataConnector=/connector/DataConnector
   ```

3.10 Configure the ece Scripts

To configure the **ece** shell script you need to create two configuration files, one for each of the Tomcat instances you created.

On each **engine-host** while logged in as **root**:

1. Create an empty file in the **/etc/escenic/** folder **called ece-engine1.conf**:

   ```
   # touch /etc/escenic/ece-engine1.conf
   ```

2. Open **/etc/escenic/ece-engine1.conf**, and enter the following:

   ```
   ece_home=/usr/share/escenic/escenic-content-engine-7.12
   java_home=/usr/lib/jvm/java-11-openjdk-amd64
   appserver=tomcat
   tomcat_home=/opt/tomcat
   tomcat_base=/opt/tomcat-engine1
   deploy_webapp_white_list="escenic-admin escenic indexer-webservice webservice
    webservice-extensions resolver plugin-apps"
   ```

   where *plugin-apps* is zero or more of the following names depending on which plug-ins you have installed:

   poll-ws
   newsgate-webservice
   live-center-presentation-webservice
   live-center-editorial
   menu-webservice
   video-presentation-webservice

3. Create an **ece-search1.conf** file by copying **ece-engine1.conf**:

   ```
   # cp /etc/escenic/ece-engine1.conf /etc/escenic/ece-search1.conf
   ```

4. Open **ece-search1.conf** in an editor, and enter the following:

   ```
   ece_home=/usr/share/escenic/escenic-content-engine-7.12
   java_home=/usr/lib/jvm/java-11-openjdk-amd64
   appserver=tomcat
   tomcat_home=/opt/tomcat
   tomcat_base=/opt/tomcat-search1
   deploy_webapp_white_list="solr indexer-webapp"
   ```

5. These are the minimum settings you need in these files, and are suitable for a personal development installation. For a production installation, you should add at least two more settings to **ece-engine1.conf**:

   ```
   escenic_admin_http_user=username
   ```

```
escenic_admin_http_password=password
```

It is possible to add even more parameter settings to achieve various objectives, but if you have followed the instructions earlier in this manual, then the above settings should be sufficient to get started. The **/etc/escenic/** folder contains a sample **ece** configuration file called **ece.conf** that contains examples of all possible **ece** configuration parameters, with explanatory comments.

6. Create a standard installation directory structure by entering the following commands:

```
# mkdir -p /var/{backups,cache,crash,lib,log,run,spool}/escenic
# chown escenic:escenic /var/{backups,cache,crash,lib,log,run,spool}/escenic -R
```

These commands will create the following folders and assign them to the **escenic** user:

/var/backups/escenic
/var/cache/escenic
/var/crash/escenic
/var/lib/escenic
/var/log/escenic
/var/run/escenic
/var/spool/escenic

> If, for any reason, you do not want to install Content Store files in standard locations and you have modified any of the following settings in your **.conf** files:
>
> **cache_dir**
> **log_dir**
> **pid_dir**
> **heap_dump_dir**
>
> then you must make sure that all the referenced folders exist and that the **escenic** user has write permission to them. You are strongly advised, however, not to do so. All instructions in this manual and the Server Administration Guide assume files are installed in standard locations.

3.11 Deploy the Built-in Applications

The Content Store includes a number of built-in web applications used for various purposes. These applications need to be deployed to the two Tomcat instances you have configured. Exactly which applications are deployed to each instance are determined by the **deploy_webapp_white_list** configuration parameter described in section 3.10.

To deploy the instances you have created, enter the following commands on each **engine-host**, while logged in as **escenic**:

```
$ ece -i engine1 -t engine deploy
$ ece -i search1 -t search deploy
```

Once you have done this, it should be possible to run your Content Store instance(s). The following section describes a few final verification and setup tasks.

3.12 Finishing Up

You should now check that everything is installed correctly, and optionally install a daemon script for running the Content Store. These tasks are described in the following sections.

3.12.1 Verify the Installation

It should now be possible to start the Content Store and verify that the major components are working. To do so, carry out the following procedure on each **engine-host**, while logged in as **`escenic`**:

1. Start the Content Store's Tomcat instances by entering:

   ```
   $ ece -i engine1 -t engine run
   $ ece -i search1 -t search run
   ```

2. Verify that both instances are running by entering:

   ```
   $ ece -i engine1 -t engine status
   $ ece -i search1 -t search status
   ```

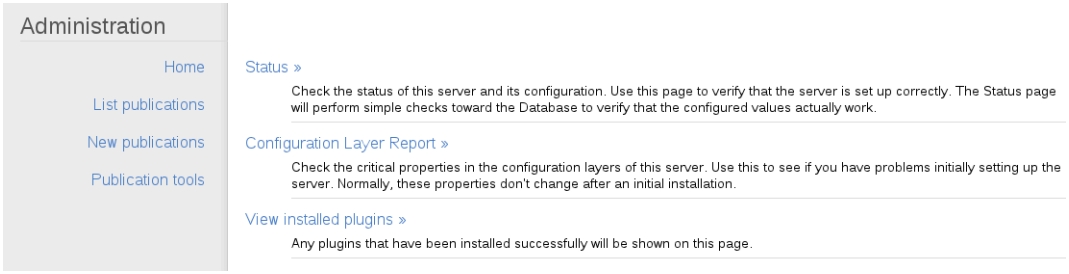
   This should produce something like the following output for each command:

   ```
   UP 0d 0h 6m 36s
   ```

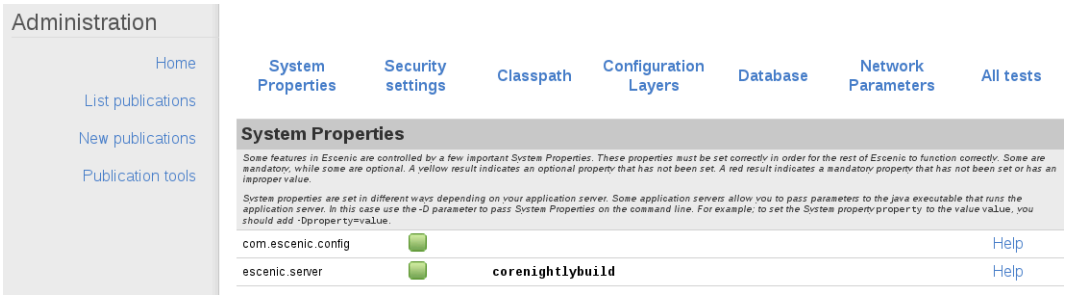
3. Open a browser, and try accessing the **`escenic-admin`** web application on each of the **engine-host**s by entering:

   ```
   http://engine-host-ip-address:8080/escenic-admin/
   ```

   in the browser address field. There may be a delay while the Content Store initializes, but eventually, you should see the following page:



4. Click on the **Status** link to display a set of status pages containing Content Store test results.



5. Across the top of these pages is a menu containing links to each of the status pages. Click on each of these links in turn and verify that all the tests pass (indicated by a icon).

6. The icon indicates that something is not correctly configured. If you see one of these icons, then you need to check over all relevant settings. In most cases there is a help link next to the test

result indicating the most likely causes of failure. Make any necessary corrections, then restart the Content Store instance by entering:

```
$ ece -i engine1 -t engine restart
```

7. Wait a minute or so, then refresh the browser window to re-execute the tests that failed.

Verify that the Content Store is correctly configured on **all** your **engine-host**s before proceeding any further.

> On some of the test pages you will see ⚠ warning icons. In most cases these are simply an indication that no publications have yet been created, and they can be ignored.

3.12.2 Configure the Daemon Script

At this point you have successfully installed the Content Store and confirmed that it runs. You also need to configure the daemon script responsible for automatically starting and stopping the Content Store together with the operating system, and test that it works. To set up the daemon script on each **engine-host**:

1. Log in as user **`escenic`**.
2. Make sure the Content Store instances are stopped.

   ```
   $ ece -i engine1 -t engine stop
   $ ece -i search1 -t search stop
   ```

3. Switch user to **`root`**.
4. Open **`/etc/default/ece`**, and find the following two lines:

   ```
   engine_instance_list=""
   search_instance_list=""
   ```

5. Edit them as follows:

   ```
   engine_instance_list="engine1"
   search_instance_list="search1"
   ```

   and save your changes.

6. Start Content Store using the daemon script:

   ```
   # /etc/init.d/ece start
   ```

   If this command does not successfully start the Content Store, then you need to fix the problem and then try again.

7. If the command does successfully start the Content Store, then stop it using the daemon script:

   ```
   # /etc/init.d/ece stop
   ```

   and enter the following command to install it as a daemon:

   ```
   # update-rc.d ece defaults
   ```

   You should now be able to reboot the **engine-host**, and expect the Content Store to be started automatically. It will also be stopped automatically when you power the **engine-host** down.

> Note that this procedure for running the daemon script is specific to Debian-based Linux distributions such as Ubuntu.

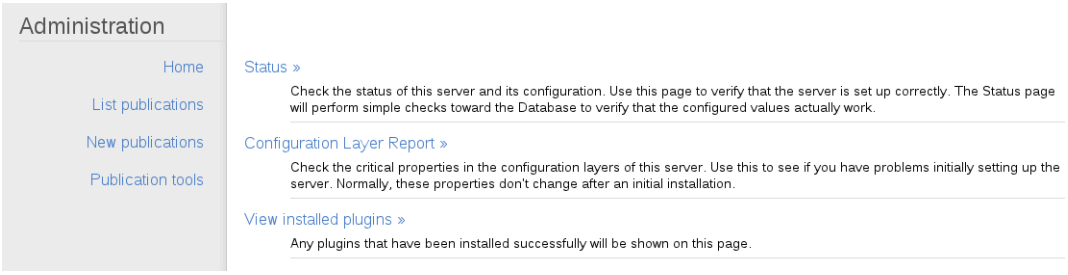
3.12.3 Create a Publication

To really verify that everything is working correctly, you need a publication in the database. The quickest way to achieve this is to add one of the demo publications included with the installation. To add the **`demo-temp-dev`** publication (which is a very simple, stripped down publication with almost no content):

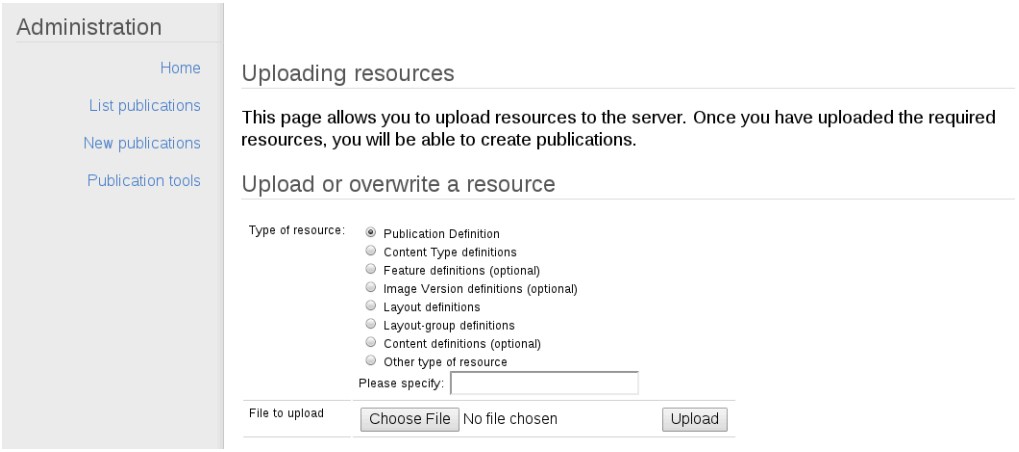
1. Download the publication WAR file from one of your **engine-host**s to the machine on which you are working. You will find **`demo-temp-dev.war`** in the **`/usr/share/escenic/escenic-content-engine-7.12/contrib/wars`** folder.
2. Open a browser, and access the **`escenic-admin`** web application on one of your **editorial-host**s by entering:

   ```
   http://editorial-host-ip-address:8080/escenic-admin/
   ```

   in the browser address field. The following page is displayed:

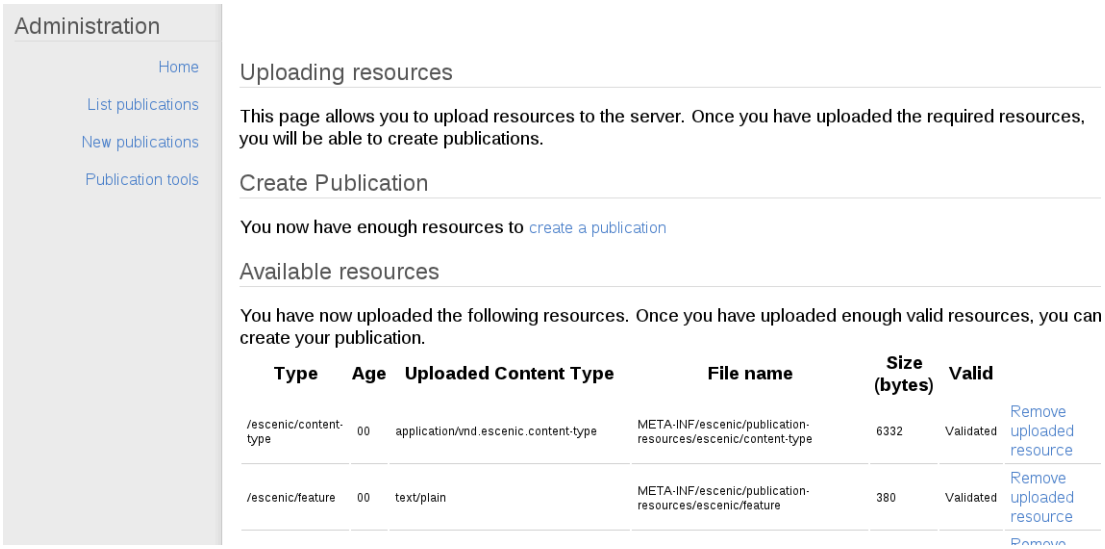


3. Click on **New publications**. The following page is displayed:

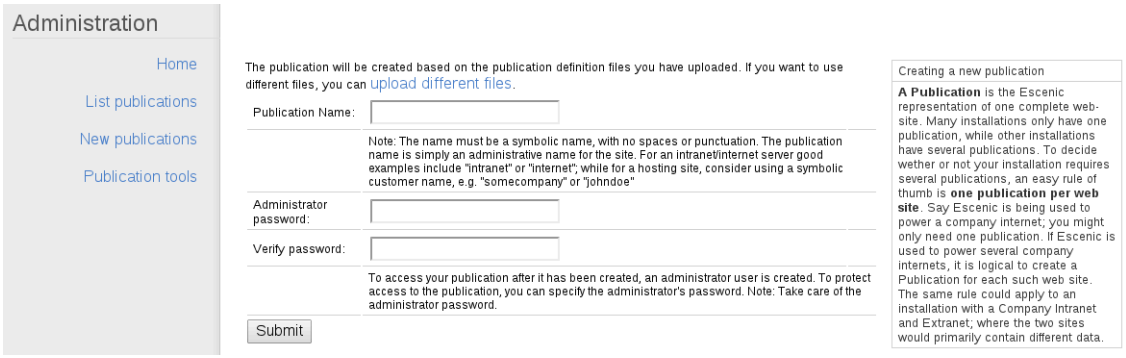


4. Click on the **Choose File** button and locate the **`demo-temp-dev.war`** file you have copied to your local machine.

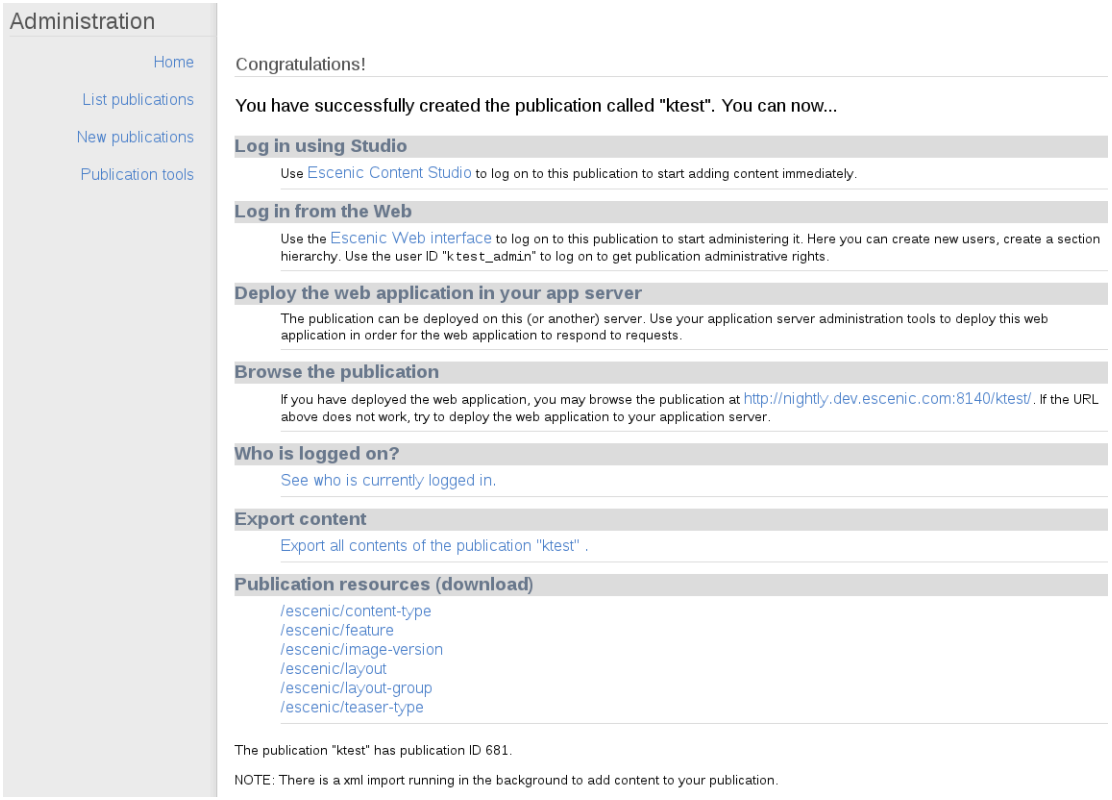
5. Click on **Upload**. The following page is displayed:



6. Click on the **create a publication** link. The following page is displayed:



7. Enter a name for the publication and an administrator password in the displayed form, then click on **Submit**. The following page is displayed:



You have now created a publication. If you click on **Home** and then **Status** to redisplay the status pages, you should see that all the ! warnings that were displayed earlier have now been replaced by icons. This should be the case not only on the **editorial-host** where you are currently working, but on all your **engine-host**s.

3.12.4 Test Web Studio

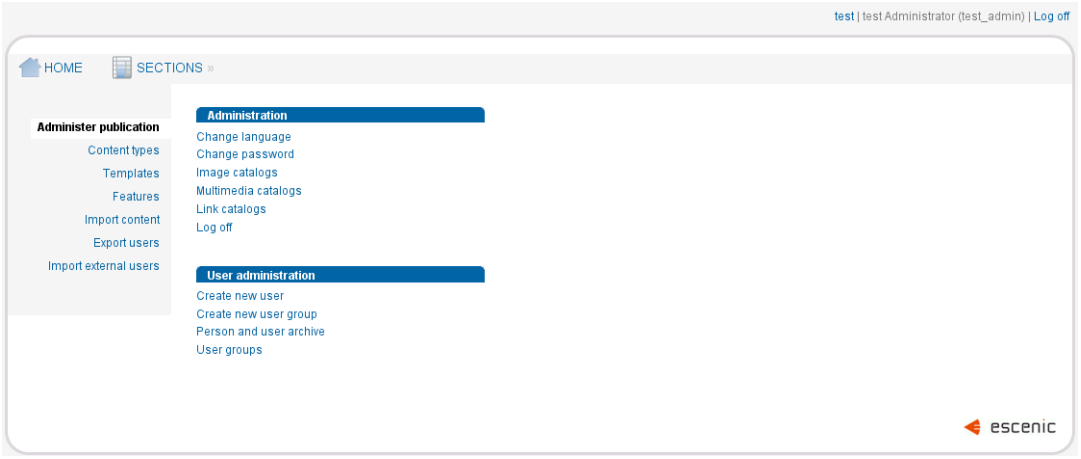
Now you have created a publication, you can use it to check that the Web Studio web application is correctly installed. To do this:

1. Open a browser, and access the Web Studio web application (called **`escenic`**) on one of your **editorial-host**s by entering:

   ```
   http://editorial-host-ip-address:8080/escenic/
   ```

   in the browser address field. A login page should be displayed.
2. Enter the **`demo-temp-dev`** publication's administrator user name (*publication-name*`_admin`) and the administrator password you specified when creating the publication.

3. The following page should then be displayed:



4 Recommended Configurations

This chapter contains detailed descriptions of recommended system configurations for the Content Store, CUE and an SSE Proxy, which is required to forward Server-Sent Events from the Content Store to CUE clients. These components can be configured to work together in a variety of ways to meet different requirements, and typical system configurations will often also involve the configuration and use of other components such as web servers, load balancers and so on. This chapter only describes two possible configurations:

- A minimal HTTP-based development/test configuration, where all components are installed on the same host
- A more robust HTTPS-based production configuration where the CUE editor is installed on a different host from the Content Store and the SSE Proxy.

These descriptions should provide a basis for understanding how the components communicate, so that you can make any adjustments needed to meet your specific requirements.

All configurations involve at least the following tasks:

- Installing an SSE Proxy (see the SSE Proxy documentation for details).
- Configuring your Content Store installation's Tomcat server to direct SSE messages to the SSE Proxy.
- Configuring the SSE Proxy to accept messages from the Content Store, and forward them to CUE.
- Configuring CUE to connect to the Content Store.

If your installation includes more than one Content Engine web service host, with a load balancer distributing requests between them, then you must ensure that both CUE and its SSE proxy use sticky sessions and connect to the same back end.

4.1 A Development/Test Configuration (HTTP)

The three main components in this configuration all belong to the host **`editorial.mydomain.com`**:

All network connections are made using HTTP. The configuration details for the Content Store, CUE and the SSE Proxy are as described in the following sections.

4.1.1 Content Engine / Tomcat Configuration (HTTP)

You need to create a connector in Tomcat for the SSE Proxy It must use a different port from the default connector used by CUE (port 8082 instead of 8080, for example). In this case CUE will connect to the Content Store on **`http://editorial.mydomain.com`** so you can set up the new connector as follows:

```
<Connector port="8082" protocol="HTTP/1.1"
  maxThreads="50" URIEncoding="UTF-8"
  SSLEnabled="false"
```

```
    proxyName="editorial.mydomain.com" proxyPort="80"
   scheme="org.apache.coyote.http11.Http11NioProtocol"/>
```

Restart Tomcat for the change to take effect.

> If you want to use Varnish with your installation, then you should set **proxyPort** to **81** and also use port **81** in your CUE configuration (see section 4.1.2).

4.1.2 CUE Configuration (HTTP)

CUE Configuration involves configuring both CUE itself and the **nginx** web server used to host it (for more about this, see the CUE documentation). The CUE configuration file **/etc/escenic/cue-web-*version*/config.yml** must contain the following setting identifying the Content Store web service to connect to:

```
endpoints:
  escenic: "http://editorial.mydomain.com:80/webservice/index.xml"
```

> If you want to use Varnish with your installation, then you should use port **81** rather than port **80** and also set **proxyPort** to **81** in your Tomcat configuration (see section 4.1.1).

The required **nginx** configuration is as follows:

- In **/etc/nginx/sites-available/default**:

  ```
  server {
    listen 80 default;
    include /etc/nginx/default-site/*.conf;
  }
  ```

- In **/etc/nginx/default-site/cue-web.conf**:

  ```
  location /cue-web/ {
          alias /var/www/html/cue-web/;
          expires modified +310s;
  }
  ```

- In **/etc/nginx/default-site/webservice.conf**:

  ```
  location ~ "/(escenic|studio|webservice|webservice-extensions)/(.*)" {
      proxy_set_header Host $http_host;
      proxy_pass http://127.0.0.1:8082;
  }
  ```

- In **/etc/nginx/conf.d/request-entity-size-limit.conf**:

  ```
  client_max_body_size 0;
  ```

4.1.3 SSE Proxy Configuration (HTTP)

To configure an SSE Proxy, you edit its **sse-proxy.yaml** configuration file. All you need to do is add the URL of the Content Store's change log SSE endpoint to it's list of backends:

```
backends:
  - uri: http://editorial.mydomain.com:8082/webservice/escenic/changelog/sse
```

Even though your SSE Proxy is in this case installed on the same host as the Content Store, you must still use the host name specified in the Tomcat connector definition: do not replace it with **localhost**.

For full information about installing and configuring an SSE Proxy, see the SSE Proxy documentation.

> Once your SSE Proxy is configured, you will also need to reconfigure the change log event stream URI that the Content Store offers to clients. By default the Content Store sends clients the URI of its own SSE endpoint, but now you want clients to get their change log events from the SSE Proxy instead, so you need to set this configuration to:
>
> ```
> sseEndpoint=http://editorial.mydomain.com:9080/
> ```
>
> For detailed instructions on how to do this, see Changing the Content Engine Event Stream URI.

4.2 A Production Configuration (HTTPS)

In this production configuration, the CUE application is installed on a different host from the Content Store and the SSE Proxy. All network communication is SSL-protected. Since the SSE Proxy does not have built-in SSL support, it is placed behind an **nginx** web server to provide that support.

In order to set up an HTTPS configuration, you need to install SSL certificates, which must be obtained from a certificate authority such as Let's Encrypt, Thawte or Verisign. There is a list of certificate suppliers here. Follow the supplier's instructions for installing the certificates they provide.

4.2.1 Content Engine / Tomcat Configuration (HTTPS)

In this case the connector you set up for the SSE Proxy must be configured to support SSL connections as follows:

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
  maxThreads="50" URIEncoding="UTF-8"
  SSLEnabled="false" secure="true"
  proxyName="cue.mydomain.com" proxyPort="443" scheme="https" />
```

Restart Tomcat for the change to take effect.

4.2.2 CUE Configuration (HTTPS)

CUE Configuration involves configuring both CUE itself and the **nginx** web server used to host it (for more about this, see the CUE documentation). The CUE configuration file **/etc/escenic/cue-web-*version*/config.yml** must contain the following setting identifying the Content Store web service to connect to:

```
endpoints:
  escenic: "https://cue.mydomain.com/webservice/index.xml"
```

The required **nginx** configuration is as follows:

- In **/etc/nginx/sites-available/default**:

  ```
  server {
    include /etc/nginx/certs/*.conf;
    include /etc/nginx/default-site/*.conf;
  }
  ```

- In **/etc/nginx/default-site/cue-web.conf**:

```
location /cue-web/ {
        alias /var/www/html/cue-web/;
        expires modified +310s;
}
```

- In **/etc/nginx/default-site/webservice.conf**:

```
location ~ "/(escenic|studio|webservice|webservice-extensions)/(.*)" {
    proxy_set_header Host $http_host;
    proxy_pass http://editorial.mydomain.com:8443/;
}
```

- In **/etc/nginx/conf.d/request-entity-size-limit.conf**:

```
client_max_body_size 0;
```

- In **/etc/nginx/certs/cue.mydomain.com.conf**:

```
listen 443 ssl;
server_name cue.mydomain.com
ssl_prefer_server_ciphers On;
ssl_certificate /etc/nginx/certs/cue.mydomain.com.pem;
ssl_certificate_key /etc/nginx/certs/cue.mydomain.com.key;
ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
ssl_ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:RSA
+AESGCM:RSA+AES:!aNULL:!MD5:!DSS;
```

  This configuration is based on the assumption that you have obtained and installed a valid SSL certificate. For information on how to do this, please see the documentation of your chosen certificate supplier.

4.2.3 SSE Proxy Configuration (HTTPS)

Add the URL of the Content Store's change log SSE endpoint to the SSE Proxy's list of backends and set the Host header as follows:

```
backends:
  - uri: http://editorial.mydomain.com:8443/webservice/escenic/changelog/sse
      httpHeaders:
        Host: cue.mydomain.com
```

For full information about installing and configuring an SSE Proxy, see the SSE Proxy documentation.

The web server in front of the SSE Proxy must be able to handle an SSE stream without constantly disconnecting clients: It must be configured to:

- Allow long-running connection
- Support chunked encoding
- Not buffer or cache the traffic

The following example shows a suitable **nginx** configuration (**/etc/nginx/default-site/sse-proxy.conf**) for a virtual host fronting an SSE Proxy installed on the same host as the Content Store:

```
location /sse {

    proxy_read_timeout     300;
    proxy_connect_timeout  300;
```

```
    proxy_pass http://editorial.mydomain.com:9080;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For $remote_addr;
    proxy_set_header Connection '';
    proxy_http_version 1.1;
    proxy_buffering off;
    proxy_cache off;
    chunked_transfer_encoding off;
  }
```

Once your SSE Proxy is configured, you will also need to reconfigure the change log event stream URI that the Content Store offers to clients. By default the Content Store sends clients the URI of its own SSE endpoint, but now you want clients to get their change log events from the SSE Proxy instead, so you need to set this configuration to:

```
sseEndpoint=https://cue.mydomain.com/sse
```

For detailed instructions on how to do this, see Changing the Content Engine Event Stream URI.

4.3 CORS Configuration

CORS (Cross-Origin Resource Sharing) is required in situations where the CUE editor is located in a different domain from other resources to which the CUE editor client will require access. Neither of the recommended configurations described in this chapter require use of CORS. Other possible configurations, however, may require some components to be configured to support CORS.

If, for example, the production configuration described in section 4.2 is modified by placing an **`nginx`** proxy in front of the Content Store web services, then a **`/etc/nginx/default-site/cors.conf`** file would need to be included in the proxy server's configuration, with the following content:

```
location ~ "/(escenic|studio|webservice|webservice-extensions)/(.*)" {
    if ($http_origin ~* (https?://[^/]*.mydomain.com(:[0-9]+)?)$) {
        set $cors "true";
    }
    if ($request_method = 'OPTIONS') {
        set $cors "${cors}options";
    }
    if ($request_method = 'GET') {
        set $cors "${cors}get";
    }
    if ($request_method = 'HEAD') {
        set $cors "${cors}get";
    }
    if ($request_method = 'POST') {
        set $cors "${cors}post";
    }
    if ($request_method = 'PUT') {
        set $cors "${cors}post";
    }
    if ($request_method = 'DELETE') {
        set $cors "${cors}post";
    }
    if ($cors = "trueget") {
        add_header "Access-Control-Allow-Origin" "$http_origin" always;
        add_header "Access-Control-Allow-Credentials" "true" always;
```

```
        add_header "Access-Control-Expose-Headers" "Link,X-ECE-Active-
Connections,Location,ETag,Allow" always;
    }
    if ($cors = "truepost") {
        add_header "Access-Control-Allow-Origin" "$http_origin" always;
        add_header "Access-Control-Allow-Credentials" "true" always;
        add_header "Access-Control-Expose-Headers" "Link,X-ECE-Active-
Connections,Location,ETag" always;
    }
    if ($cors = "trueoptions") {
        add_header 'Access-Control-Allow-Origin' "$http_origin";
        add_header 'Access-Control-Allow-Credentials' 'true';
        add_header 'Access-Control-Max-Age' 1728000;
        add_header 'Access-Control-Allow-Methods' 'GET, POST, HEAD, OPTIONS, PUT,
 DELETE';
        add_header 'Access-Control-Allow-Headers' 'Authorization,Content-
Type,Accept,Origin,User-Agent,DNT,Cache-Control,X-Mx-ReqToken,Keep-Alive,X-Requested-
With,If-Modified-Since,If-Match,If-None-Match,X-Escenic-Locks,X-Escenic-media-
filename,X-Escenic-home-section-uri';
        add_header 'Content-Length' 0;
        add_header 'Content-Type' 'text/plain charset=UTF-8';
        return 204;
    }
}
```

5 Recommended Modifications

This section contains instructions for various additions and modifications you can make to the basic installation described in chapter 3. These changes are recommended for production installations.

5.1 Search Engine Deployment and Configuration

The Content Store's search functionality is provided by the Java search engine Apache Solr, which runs as a web application. The standard Content Store installation includes a **`solr`** web application and an associated **`indexer`** web application that indexes the content of all CUE publications. These two applications are deployed along with the Content Store by the **`ece`** script's **`deploy`** action.

The result of following the basic installation procedure described in chapter 3, therefore, is that a **`solr`** instance and **`indexer`** web application is deployed on every **engine host** in your installation, all with identical configurations.

This set up will work, but it is relatively inefficient and is unlikely to work well in a production environment. There are two main reasons for this:

Solr memory usage
`solr` and the **`indexer`** can at times consume large amounts of memory and trigger large garbage collection operations in the JVM, which has severe effects on Content Store performance. They should therefore not be run in the same JVM as the Content Store on production systems. **`solr`** already runs in its own webapp container (and therefore in a different JVM), but the indexer is deployed to the same Tomcat instance as the Content Store. The simplest way to achieve this separation on a single-host installation is to move the **`indexer`** webapp to a separate Tomcat instance. For more about this, see Isolating The Search Engine.

Solr stemming
In the default **`solr`** configuration, English stemming is enabled by default. This means that searching non-English content might give unexpected results.

If your content is in a language other than English, you should either disable stemming or modify the configuration to suit your language.

To disable stemming, remove the following line from **`schema.xml`**:

```
<filter class="solr.SnowballPorterFilterFactory" protected="protwords.txt"
 language="English"/>
```

Disabling stemming will improve **`solr`**'s performance.

For information about how to configure stemming for other languages, see the Solr documentation on http://lucene.apache.org/solr/.

Solr optimization issues
The default **`solr`** configuration is optimized for editorial purposes: it indexes all the fields needed to support the search functionality provided by CUE, resulting in very large indexes. This is acceptable in the editorial context, since the number of concurrent CUE users, even in a very large organisation, is not likely to be very large. The **presentation host**s in a large CUE

installation, however, can be required to serve many thousands of concurrent users, and the default **`solr`** configuration may perform poorly in this context.

The default configuration, therefore, is fine for the **editorial host**s in a production system, but for the **presentation host**s you are recommended create a custom indexer configuration that only indexes the fields actually needed to support the kinds of search required in your publications.

To do this, open **`/var/lib/escenic/`***solr-core*`**/schema.xml`** for editing on each of your **presentation host**s, and modify the index schema to meet your requirements. Editing this file is outside the scope of this manual. In order to tune the search engine you need to take account of the both the contents of your publications, your users' needs with regards to search and the limitations imposed by your particular hardware configuration. For further information and advice on tuning, see the Solr documentation on http://lucene.apache.org/solr/.

There are many more changes you can make to your search engine set-up in order to optimize it for your particular needs. For a discussion of the general principles involved, see Search Engine Configuration and Management.

5.2 Password Protect escenic-admin

escenic-admin is a web application for administration of the Content Store. In a default installation it is installed without password protection. **`escenic-admin`** is, however, a powerful tool and should definitely not be openly accessible in a production environment. You are therefore strongly recommended to limit access to a small group of competent users.

To password protect **`escenic-admin`**:

escenic_admin_http_user=

escenic_admin_http_password=

1. Set **`escenic-admin-authentication = true`** in the **`assemblytool`** properties file before running the **`assemblytool`**.
2. Define one or more user log-ins for **`escenic-admin`** in Tomcat's **`tomcat-users.xml`** file.

To define an **`escenic-admin`** user in Tomcat, you need to add an entry like this to **`tomcat-users.xml`**:

```
<user username="admin-user" password="admin-password" roles="ECEAdmin"/>
```

where:

- *admin-user* is the user name you want to use for logging in to **`escenic-admin`**
- *admin-password* is the password you want to use for logging in to **`escenic-admin`**

> It is obviously **not** a good idea to enter the password in this file as plain text. For a description of how to encrypt the password, see http://www.jdev.it/encrypting-passwords-in-tomcat/.

6 Upgrading

All you need to do to upgrade your Content Store (including any plug-ins that you have installed from APT/RPM packages) is:

1. Read the release notes for your planned upgrades (both Content Store and plug-ins). Make a note of any special tasks that need to be carried out in connection with the upgrades. Some changes, for example, require you to run a database upgrade script after the upgrade.
2. Log in as **`root`** on each of your **engine-hosts**, and enter the following commands:

   ```
   # apt-get update
   # apt-get upgrade
   ```

3. Carry out any required upgrade tasks. For general information about how to carry out a database upgrade should that be necessary, see: section 6.1.

6.1 Database Upgrades

If the upgrade tasks for the upgrade you are carrying out include upgrading the database, then a script for performing the upgrade will be included in the distribution. All you need to do is log in as the **`escenic`** user on your **database-host** and run the appropriate script for your particular database. The scripts are located in the **`/usr/share/escenic/escenic-content-engine-7.12/contrib/database/sql/`** folder. Here you will find a subfolder for each supported database - **`MySQL`** and so on (**`MySQL`** = MariaDB). In each subfolder is an **`upgrade`** folder and a script for upgrading to the current version (plus scripts for upgrading to a number of previous versions. To upgrade a MariaDB database, for example, you would need to run:

```
/usr/share/escenic/escenic-content-engine-7.12/contrib/database/sql/MySQL/upgrade/
to-version
```

where *version* is the current Content Store version number.