# Table of Contents

Software Requirements .................................................................................................................. 1  
Deployment Overview .................................................................................................................... 2  
  Servlet container or Fusion Web Apps service ............................................................................. 2  
  Standalone deployments .............................................................................................................. 2  
Deployment in a Servlet Container .................................................................................................. 3  
  Create a WAR file ....................................................................................................................... 4  
  Deploy the WAR file ................................................................................................................... 5  
Fusion Web App Deployment ......................................................................................................... 6  
  UI deployment overview ............................................................................................................. 7  
  Publishing a search interface to Fusion Server ............................................................................ 8  
  Migrating a search interface to other Fusion hosts ..................................................................... 10  
    Download or build a WAR file ............................................................................................... 10  
    Downloading a WAR file ....................................................................................................... 10  
    Building a WAR file .............................................................................................................. 12  
    Upload the WAR file ............................................................................................................. 13  
    Uploading with the Fusion UI ............................................................................................... 13  
    Uploading with the REST API .............................................................................................. 15  
Running a dedicated node for a search interface .......................................................................... 16  
Standalone Deployments ............................................................................................................... 17  
  Embedded Tomcat web server ................................................................................................. 18  
  Standalone app directory ......................................................................................................... 18  
  Executable Java JAR file .......................................................................................................... 19  
  Running an executable JAR file ............................................................................................... 20  
Building an App from Source ...................................................................................................... 21  
  Package for deployment to a servlet container ....................................................................... 22  
  Package as a standalone app .................................................................................................... 23  
    Standalone app directory ..................................................................................................... 23  
SSL Configuration ......................................................................................................................... 24  
  SSL keystore ............................................................................................................................ 25  
  SSL parameters ....................................................................................................................... 26  
Encrypting Sensitive Values ......................................................................................................... 27  
Enabling Social Features .............................................................................................................. 28  
  Configure Fusion ...................................................................................................................... 29  
  Configure App Studio ............................................................................................................... 31  
    Upgrade to the latest version of Appkit .............................................................................. 31  
    Enable social features .......................................................................................................... 31
**Software Requirements**

This software is required to use App Studio and Appkit to develop, build, and deploy search apps.

<table>
<thead>
<tr>
<th>Software</th>
<th>Version and Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appkit</td>
<td>A version of Appkit is included in each App Studio release. Appkit is also released independently. For information about upgrading an app to use a later version of Appkit, see Upgrade Appkit in existing apps.</td>
</tr>
<tr>
<td>Oracle Java 8 JDK or above</td>
<td>Download and install the Oracle Java SE JDK 8 or newer for Windows or Linux (it comes pre-installed on Macs).</td>
</tr>
<tr>
<td>Node.js 8 or above</td>
<td>Download and install Node.js 8 or above.</td>
</tr>
<tr>
<td>npm</td>
<td>The npm package manager is included with Node.js.</td>
</tr>
</tbody>
</table>
Deployment Overview

You can choose how to deploy your search application:

- **Servlet container or Fusion Web Apps service** – Deploy your search application in a servlet container or in the Fusion Web Apps service.

<table>
<thead>
<tr>
<th>Tip</th>
<th>This is the recommended deployment method for production.</th>
</tr>
</thead>
</table>

  - Deploy your search application as a standalone application.

**Servlet container or Fusion Web Apps service**

Deploy your search application in a servlet container:

- **Java WAR file for deployment in a servlet container**

  Package your search app in a Java WAR file, which can be deployed in a servlet container such as Tomcat or Jetty. This is the most common way to deploy a search app.

- **Fusion Web app deployment** gives you an embedded instance of App Studio in each of your Fusion instances.

**Standalone deployments**

Two types of standalone deployments are supported:

- **Java JAR file with embedded Tomcat web server**

  Package your search app in a Java JAR file that also contains an embedded Tomcat web server. The app is ready to deploy in a production environment. It isn’t necessary to build a WAR file and deploy that in a servlet container or application server.

- **Executable Java JAR file for execution in a Java runtime environment**

  Package your search app in a Java JAR file. You can run the app with the Java launcher.
Deployment in a Servlet Container

The most common way to deploy a search application is to package it as a standard Java .war file, which can be deployed in any standard servlet container such as Tomcat or Jetty.
Create a WAR file

Create a WAR file suitable for deployment in a servlet container. Do either of the following:

- **Use a script** – Run the script `app-studio` (on Unix or MacOS) or `app-studio.bat` (on Windows) with the parameter `package` to create the WAR file.

- **Download the WAR file from Fusion** – If you created the search interface in Fusion, you can download the WAR file from Fusion.
Deploy the WAR file

Deploy the WAR file in the servlet container. Follow the instructions in the documentation for the servlet container.
### Fusion Web App Deployment

You can use App Studio to deploy a search app in Fusion Server. You develop your search interface, and then publish the search app to Fusion's Web Apps service.

<table>
<thead>
<tr>
<th>Note</th>
<th>This deployment type leverages your existing production environment, requiring no dedicated nodes for App Studio. However, you might need to deploy additional Fusion nodes if you anticipate significant traffic to your search interface.</th>
</tr>
</thead>
</table>


Here's an overview of the steps:

1. Develop your search interface.

2. Publish your UI to Fusion Server using the instructions below.

Your web app then becomes accessible at `<a href="http://localhost:8764/webapps/&lt;project-name&gt;" class="bare">http://localhost:8764/webapps/&lt;project-name&gt;</a>`. 
Publishing a search interface to Fusion Server

These steps publish your search interface to a Fusion instance (and its cluster) where the App Studio project already exists.

1. In the Fusion workspace, navigate to App Studio and select your search interface.

The configuration panel appears.

2. Next to Publish Status, click the toggle:
3. Click **Save**.

The interface is only published after you click **Save**, which also enables the **View Published UI** button.

4. Click **View Published UI** to launch a new window where you can view the interface as an end user.

Your published interface is available at [http://localhost:8764/webapps/<project-name>/#/search](http://localhost:8764/webapps/<project-name>/#/search).
Migrating a search interface to other Fusion hosts

If you’ve published your search interface on one Fusion Host in a cluster, then it is published throughout the cluster.

There are two ways to migrate a search interface to a different Fusion cluster:

<table>
<thead>
<tr>
<th>Export and import the Fusion app</th>
<th>Download and upload the WAR (.war) file</th>
</tr>
</thead>
<tbody>
<tr>
<td>This migrates all objects in the Fusion app, such as datasources, query profiles, schedules, and so on, in addition to the search interface. The search interface will be editable on the target hosts.</td>
<td>This migrates only the search interface. The search interface will not be editable on the target hosts.</td>
</tr>
<tr>
<td>For instructions, see Working With Apps in Fusion Server’s Getting Started guide.</td>
<td>For instructions on downloading or building a WAR file and uploading it to Fusion, see below.</td>
</tr>
</tbody>
</table>

Download or build a WAR file

You can distribute a search interface as a WAR (.war) file.

You can download a WAR file from the Fusion UI or create one from a downloaded project.

**Downloading a WAR file**

1. In the Fusion workspace, navigate to App Studio and select your search interface.
The configuration panel appears.

2. Click Download war:
Building a WAR file

A downloaded project comes with the scripts app-studio (for Unix) and app-studio.bat (for Windows) that create WAR files.

How to build a WAR file

Unix and MacOS:

1. In a shell window, switch to your project directory:

   ```
   cd /path/to/project-directory
   ```

2. Create application files:

   ```
   ./app-studio package
   ```

Windows:

1. In a terminal window, switch to your project directory:

   ```
   cd \path\to\project-directory
   ```

2. Create application files:

   ```
   app-studio.bat package
   ```

Application files created:
The script creates the following files in the `dist` directory of your project:

- **search-app-4.2.x.jar**
  
  This is a executable Java JAR file.

- **search-app-project.zip**
  
  Use this file to share your project with other search interface developers.

- **search-app.war**
  
  You can upload this file to Fusion to deploy it. See the Deployment Guide.

## Upload the WAR file

You can upload the WAR file through the Fusion UI or the REST API.

### Uploading with the Fusion UI

You can upload your project (as a WAR file) in the App Studio configuration panel.

| Note | If you upload the file into a Fusion app other than the one in which it was created, the new Fusion app must include a query profile and data fields whose names are identical to the ones already configured in the search interface. |

1. From the App Studio menu, select your project:
The project configuration panel appears.

2. Click *Upload*. 
Fusion prompts you to confirm that you want to upload a project that will delete the existing one.

3. Click **Yes, continue**.

4. Navigate to your WAR file and select it.

   After the upload is complete, you can edit or publish the uploaded project.

### Uploading with the REST API

Use the `webapps/{id}/war` endpoint of the Webapps API to upload a WAR file, as in this example:

```
curl -u admin:password123 -X PUT http://localhost:8764/api/webapps/Movie_Search/war -F 'file=@movies.war'
```
Running a dedicated node for a search interface

When deploying Fusion nodes dedicated to serving one or more search interfaces, only the following Fusion services need to run:

- agent
- api
- log-shipper
- webapps

When starting a dedicated node, you can minimize the number of running services by starting it like this.

**Unix and MacOS:**

```
cd {fusion-bin-unix}
./agent start
./api start
./log-shipper start
./webapps start
```

**Windows:**

```
cd {fusion-bin-windows}
agent.cmd start
api.cmd start
log-shipper.cmd start
webapps.cmd start
```
Standalone Deployments

There are two ways to deploy a standalone search interface:

- Create an app with an embedded Tomcat web server

  This deployment type has an embedded Tomcat web server and is ready to deploy in a production environment. Previously, this was referred to as a "self-contained deployment."

- Create an executable Java JAR file

  This is a JAR package. Previously, this was referred to as a "self-executing deployment."
Embedded Tomcat web server

Package your app as a standalone app for deployment to a production server. The standalone app embeds an Apache Tomcat servlet container.

Unix or MacOS:

```
./app-studio dist
```

Windows:

```
app-studio.bat dist
```

Standalone app directory

Unix or MacOS:

The standalone app for deployment is in the `app-name/search-app-standalone` directory.

```
search-app-standalone/
  README.md
  app/
  build/
  config/
  keystore.jks
  lib
  search-app-start.bat
  search-app-stop.bat
  search-app-stop.ps1
  search-app.sh
```

Windows:

The standalone app for deployment is in the `app-name\search-app-standalone` directory.

```
search-app-standalone\
  README.md
  app\n  build\n  config\n  keystore.jks
  lib
  search-app-start.bat
  search-app-stop.bat
  search-app-stop.ps1
  search-app.sh
```

The `search-app-project.zip` is a snapshot of your project at the time you compiled the application.
Executable Java JAR file

In this deployment type, your search interface runs within the Java runtime environment (JRE).

A downloaded project comes with an `app-studio` script that creates a JAR file.

How to create the JAR file

1. In a shell window, switch to your project directory:
   ```
   cd /path/to/project-directory
   ```

2. Create the JAR file:
   - **Unix and MacOS:**
     ```
     ./app-studio package
     ```
   - **Windows:**
     ```
     app-studio.bat package
     ```

The script creates the executable JAR file in the `dist` directory of your project:
```
search-app-{version}.x.jar
```
Running an executable JAR file

```java
java -jar search-app-{version}.x.jar
```
Building an App from Source

This topic explains how to build an app from source for deployment in an external servlet container or in the Fusion Web Apps service or for deployment as a standalone app.

Before building and deploying an app, check the software requirements.
Package for deployment to a servlet container

Package your app as a WAR file for deployment to an external servlet container, e.g. Apache Tomcat.

How to package your app

Unix or MacOS:

```
./app-studio package
```

The WAR file for deployment is `search-app.war` in the `app-name/dist` directory.

Windows:

```
app-studio.bat package
```

The WAR file for deployment is `search-app.war` in the `app-name\dist` directory.

### Note

If your search app is in Fusion Server, you can also package the app to a WAR file from Fusion Server.
Package as a standalone app

Package your app as a standalone app for deployment to a production server. The standalone app embeds an Apache Tomcat servlet container.

Unix or MacOS:

```
./app-studio dist
```

Windows:

```
app-studio.bat dist
```

Standalone app directory

Unix or MacOS:

The standalone app for deployment is in the `app-name/search-app-standalone` directory.

```
search-app-standalone/
   README.md
   app/
   build/
   config/
   keystore.jks
   lib
   search-app-start.bat
   search-app-stop.bat
   search-app-stop.ps1
   search-app.sh
```

Windows:

The standalone app for deployment is in the `app-name\search-app-standalone` directory.

```
search-app-standalone\
   README.md
   app\
   build\
   config\
   keystore.jks
   lib
   search-app-start.bat
   search-app-stop.bat
   search-app-stop.ps1
   search-app.sh
```
SSL Configuration

App Studio can be served over HTTPS using SSL encryption. You can use the default keystore for development and testing, or use your own keystore for production. Then, invoke the App Studio startup script using the SSL parameters.
SSL keystore

We include a keystore file with a default self-signed key for development and testing.

For proper security in a production environment, import your own keystore into the keystore.jks file, or copy it to a new file. If you copy it to a new file, use the -Dtwigkit.keystore.file (described below) to specify its location.
SSL parameters

To enable SSL, you specify the following parameters on the command line when invoking the startup script:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Dtwigkit.ssl=true</td>
<td>Enable SSL.</td>
<td>false</td>
</tr>
<tr>
<td>-Dtwigkit.https.port</td>
<td>Set the port.</td>
<td>8765</td>
</tr>
<tr>
<td>-Dtwigkit.keystore.file</td>
<td>The keystore path/filename, relative to the app-studio directory.</td>
<td>keystore.jks</td>
</tr>
<tr>
<td>-Dtwigkit.keystore.password</td>
<td>The keystore password.</td>
<td>p4ssw0rd</td>
</tr>
<tr>
<td>-Dtwigkit.keystore.alias</td>
<td>The name of the key in the keystore to be used.</td>
<td>default-key</td>
</tr>
</tbody>
</table>
Encrypting Sensitive Values

In a downloaded project, the `bin/twigcrypt/twigcrypt.sh` utility is available to encrypt sensitive string values, such as passwords, at the command line. It uses a two-way encryption mechanism so anywhere in the code this is used the value can be decrypted.

To encrypt a value, run the following (note the single quotes around `yourSensitiveValue`):

```
./twigcrypt.sh yourSecretSeed 'yourSensitiveValue'
```

This outputs an encrypted string. You must copy the whole string and paste into your configuration file.

For example, in your `src/main/resources/conf/platforms/fusion/fusion.conf` file, add:

```
username:jbloggs
password:Enc(ABC123==)
```

Then you must also configure the seed in the application's security configuration, in `src/main/resources/conf/security/security.conf`:

```
password: yourSecretSeed
```

Wherever this configuration parameter is used, it will be decrypted back to plain text at the time it's used.

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lucidworks recommends using a randomly-generated <strong>alphanumeric</strong> seed (special characters can cause problems).</td>
</tr>
</tbody>
</table>
Enabling Social Features

Prior to using social and collaboration tools, you must enable them in Fusion and in App Studio.
Configure Fusion

How to configure Fusion

1. Update the managed-schema to include the new social fields:

```xml
<field indexed="true" multiValued="false" name="type" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="user_id" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="full_name" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="anonymous" required="false" stored="true" type="boolean"/>
<field indexed="true" multiValued="false" name="created" required="false" stored="true" type="pdate"/>
<field indexed="true" multiValued="false" name="created_epoch" required="false" stored="true" type="plong"/>
<field indexed="true" multiValued="false" name="name" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="path" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="private" required="false" stored="true" type="boolean"/>
<field indexed="true" multiValued="false" name="query_url" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="collection" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="target" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="title" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="description" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="url" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="accessibility" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="namespace" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="creator__anonymous" required="false" stored="true" type="boolean"/>
<field indexed="true" multiValued="false" name="creator__full_name" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="creator__id" required="false" stored="true" type="plong"/>
<field indexed="true" multiValued="false" name="creator__user_id" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="topic__id" required="false" stored="true" type="plong"/>
<field indexed="true" multiValued="false" name="topic__title" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="topic__private" required="false" stored="true" type="boolean"/>
<field indexed="true" multiValued="false" name="topic__path" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="topic__namespace" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="topic__description" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="topic__accessibility" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="topic__collection" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="topic__created" required="false" stored="true" type="string"/>
```
<field indexed="true" multiValued="false" name="topic__created_epoch" required="false" stored="true" type="plong"/>
<field indexed="true" multiValued="false" name="topic__creator__anonymous" required="false" stored="true" type="boolean"/>
<field indexed="true" multiValued="false" name="topic__creator__full_name" required="false" stored="true" type="string"/>
<field indexed="true" multiValued="false" name="topic__creator__id" required="false" stored="true" type="plong"/>
<field indexed="true" multiValued="false" name="topic__creator__user_id" required="false" stored="true" type="string"/>
Configure App Studio

Upgrade to the latest version of Appkit

Upgrade to the latest version of Appkit.

Enable social features

1. Create the file `social.conf` in the `conf/social` directory and add the following parameter to it:

```
platform: platforms.fusion.social
```

2. Create the file `social.conf` in the `conf/platforms/Fusion` directory and add the following parameters to it:

```
# Required for query/index pipeline
collection: fusion-app-name_user_data

# Required for query/index profiles
index-profile: fusion-app-name_user_data
query-profile: fusion-app-name_user_data

# Allow Fusion platform to store Social entities
readOnly: false
webservice-enabled: false
```