KeyTerms Installation Guide

Last edited: 10 July 2018

General Information

This document describes the steps for installing the *KeyTerms* terminology management software, with detailed instructions for installing its three main components: KeyTerms server, KeyTerms client and NlpServices.

The KeyTerms installation is dependent on the installation of several third party software capabilities. It is recommended that the dependencies be installed by a professional systems administrator or dev-ops engineer. Although we provide some tips for installing the third-party components, we strongly recommend that the installation guides for the appropriate versions of those components be consulted.

For a more thorough description of KeyTerms capabilities, see the KeyTerms User Manual (in progress).

Organization of this document

Back to top

System Requirements

Back to top

The system has been tested on CentOS 6.x and 7.x. Testing has been performed with all three components deployed on the same server, but each component may be deployed on a different server. The system configuration files must be updated to accurately reflect the locations of each component, as described in the section `Server configuration file`

- **KeyTerms Server** should be deployed on a Linux operating system. KeyTerms server deployments have been tested on CentOS 6.x and CentOS 7.x.

- **KeyTerms Client** is deployable on any operating system with a web server capable of serving static NodeJS pages. KeyTerms Client has been tested on CentOS 6.x and CentOS 7.x.

- **KeyTerms NlpServices** is a web application that is deployable on any operating system with a web server capable of serving Java web applications. KeyTerms Client has been tested using Apache TomCat 7.x under Linux CentOS 6.x and CentOS 7.x., as well as Windows 10.

The installation instructions provided in this guide assume that the three core KeyTerms components are all installed on the same server.

KeyTerms Components

Back to top

KeyTerms is comprised of three main components:

- **KeyTerms Server** - The back-end service that provides the KeyTerms functionality and serves the RESTful API. Installing the server
- **KeyTerms Client** - A set of static NodeJS pages that, when delivered from a web server, provide the KeyTerms user interface. Installing the client
- **NlpServices** - A web application that provides linguistic support to KeyTerms. This application is deployed as a java .war file via a web server such as Apache TomCat.Installing the KeyTerms NlpServices

Third-party Dependencies

Back to top

KeyTerms relies upon the previous installation of a number of third-party capabilities. These tools are often already present in an organization, and supported at the organization level. Please consult your system administrators for information on the status and usage requirements for these tools in your setting. If a fresh installation of any of these components is required, we recommend the use of a trained system administrator or dev-ops engineer. The best way to perform an install for each of these capabilities is to consult each product's documentation. We provide some tips for installation under the environments tested by the KeyTerms development team.

- **MongoDB** version 3.4 (minor version numbers acceptable)
- **Node.js** version 6.x.
- **Elastic Search** version 5.2.1
- **Tomcat 7.0 or greater** required to host NLP Services
- **Java 1.8.0 or greater** required for Tomcat and NLP Services

Installing KeyTerms

Back to top

The main steps for installing KeyTerms are:

1. Verify the availability and functionality of all of the dependencies
2. Install NlpServices
3. Verify NlpServices installation
4. Install KeyTerms server
5. Configure KeyTerms server
6. Verify KeyTerms server installation
7. Install KeyTerms client
8. Configure KeyTerms client
9. Verify KeyTerms client installation

Installing NLP Services

Back to top

NLPServices is a Java web application that comes bundled with KeyTerms. It is deployed as a .war file. Follow the installation instructions for your web server for deploying .war applications.

For example, under Apache Tomcat, place the file NLPServices .war into Tomcat's webapps directory. Restart the TomCat server. If the auto-deploy function of Tomcat is enabled, simply restarting the Tomcat server should make NLP services available. For non-standard configurations, please consult the Apache Tomcat documentation (https://tomcat.apache.org/) on deploying web applications .

Verifying NLP Services installation

Back to top

Once the NLPServices .war file has been deployed under a web application server such as Tomcat, test NLP services from any web browser with the following URL, substituting your server and port names as appropriate.

```
http://keyterms.mycompany.com:8080/NLPServices/TextServices?srctxt=马马虎虎
&lang=zho&index=false
```

You should receive a JSON response resembling the following:

```
[
 {"Order":0,"isSrcScript":true,"Script":"hani","TextIndex":"马马虎虎","Text":"马
马虎虎","TransformType":"Original Text"},
 {"Order":1,"isSrcScript":true,"Script":"hans","TextIndex":"马马虎虎","Text":"马
马虎虎","TransformType":"Simplified"},
 {"Order":2,"isSrcScript":true,"Script":"hant","TextIndex":"马马虎虎","Text":"馬
馬虎虎","TransformType":"Traditional"},
 {"Order":3,"isSrcScript":false,"Script":"latn","TextIndex":"ma ma hu
hu","Text":"mă mă hŭ hŭ","TransformType":"Pinyin"},
 {"Order":4,"isSrcScript":false,"Script":"latn","TextIndex":"ma ma hu
hu","Text":"ma3 ma3 hu3 hu3","TransformType":"Wade-Giles"}
]
```

Installing KeyTerms Server

Back to top
Back to top

Steps for installing the KeyTerms server include:

1. **Verify dependencies**
2. **Unpack deployment files**
3. **Uppdate NodeJS module dependencies (optional)**
4. **Edit server configuration file**
5. **Set up database for first use**

Unpacking the deployment files

Back to top

KeyTerms is delivered as a compressed archive. The recommended way to deploy and update KeyTerms is to create date-stamped directories for the client and the server under the $KEYTERMS_HOME base directory. Symbolic links can then be created that point to the most recent directory. The symbolic link files can be referenced in scripts for starting and stopping the servers. This has the benefit of making updates simpler, because subsequent installations will merely involves deleting and recreating the symbolic link to point to the new server and client directories.

The steps described in this section assume that you are deploying the KeyTerms client, server and NlpServices on the same machine. Make appropriate adjustments if the components are deployed on different machines.

```
../$KEYTERMS_HOME
  Server.KeyTerms-->server.keyterms.YYYY-MM-DD
  Client.KeyTerms-->client.keyterms.YYYY-MM-DD
  config.js
  ./archive
  ./certs
  ./server.keyterms.YYYY-MM-DD
  ./client.keyterms.YYYY-MM-DD
```

Note: The first time you install KeyTerms, config.js will not exist under $KEYTERMS_HOME. See server configuration for recommendations regarding the server configuration file. Similarly, the 'archive' directory is an optional directory that you may create for temporarily storing previous installations while new versions are being configured.

Steps for unpacking and setting up the distribution:

1. Create $KEYTERMS_HOME if it does not exist, for example:

```
cd  /usr/local/
mkdir keyterms
export KEYTERMS_HOME=/usr/local/keyterms
```

2. Unpack the keyterms distribution file into $KEYTERMS_HOME. Unzipping the deployment file should produce the current installation's date-stamped client and server directories. i.e. server.keyterms.YYYY-MM-DD, and client.keyterms.YYYY-MM-DD.

```
cd  $KEYTERMS_HOME
unzip keyterms_deploy_2017-03-21.zip
```

3. Set up non-root users to run the nodejs and tomcat processes that will serve the three main KeyTerms components.

```
groupadd keyterms
useradd -M -G keyterms nodejs
useradd -M -G keyterms tomcat
```

4. Change the ownership of the newly installed client and server directories

```
cd $KEYTERMS_HOME
chown -R nodejs:keyterms server.keyterms.YYYY-MM-DD
chown -R nodejs:keyterms client.keyterms.YYYY-MM-DD
```

5. Create symbolic links for the client and the server if they don't already exist

```
cd  $KEYTERMS_HOME
ln -s $KEYERMS_HOME\server.keyterms.YYYY-MM-DD Server.KeyTerms
ln -s $KEYERMS_HOME\client.keyterms.YYYY-MM-DD Client.KeyTerms
```

6. Set environment variables for the base directories of the client and the server

```
export KTSERVER_BASE=$KEYTERMS_HOME/Server.KeyTerms
export KTCLIENT_BASE=$KEYTERMS_HOME/Client.KeyTerms
```

Updating NodeJS module dependencies

Back to top

This is an optional step that is not normally recommended. If you are deploying on a system with no internet connection, this update step should be unnecessary, since compatible versions of each module are packaged with the KeyTerms distribution. It is recommended to perform updates only as necessary, since updates may cause unexpected system behavior. If you decide to update the NodeJS module dependencies, issue the following commands:

```
cd  $KTSERVER_BASE/dist
npm install
```

Server configuration

Back to top

KeyTerms server comes with a configuration file (`config.js`). KeyTerms supports the retention of previous configuration settings when the software is updated by using a two file system, where a config.js file located in a directory above that of the unpacked directory of KeyTerms server.

For expository purposes, assume that all components and/or versions of KeyTerms are routinely installed in a directory referred to as $KEYTERMS_HOME. KeyTerms Server can then be unpacked inside of $KEYTERMS_HOME, and the resulting directory referred to se $KTSERVER_BASE. Under this setip, the server configuration file shipped with the deployment is located in `$KEYTERMS_HOME/$KTSERVER_BASE/config.js`. However, if a second config file exists one level up, at `$KEYTERMS_HOME/config.js`, then this config file will be used by the server at run time. This allows configuration settings from previously deployed KeyTerms instances to be re-used.

It is important to note that even if a config.js file exists in a higher level directory than that of the server application, e.g. $KEYTERMS_HOME, a config.js file nonetheless needs to exist in the server application directory, e.g. KTSERVER_BASE. Without a config.js in the server application directory, KeyTerms server will not function. **~~

Note: It is strongly advised to use absolute paths in the KeyTerms configuration file in order to avoid potential errors caused by the way Nodejs "dynamically" generates absolute paths via `__dirname`, which will resolve to the absolute path of directory which contains the file currently executing rather than to the base installation directory. Review the Node.js documentation for more information on this behavior.

If deploying under SSL, please note that by default, KeyTerms will look for certs at the same level as the executing `config.js` file. Using the above example, if two config.js files exist, in both $KEYTERMS_HOME and $KTSERVER_BASE, then KeyTerms will attempt to read certs from `$KEYTERMS_HOME/certs/`
because `$KEYTERMS_HOME/config.js` is the config that ends up "executing". Therefore it is also highly recommended to either use absolute paths within `config.js` to specify the location of your certs *or* to keep your `certs` directory at the same level as your preferred location for `config.js`.

Server configuration settings

Back to top

Multiple configuration settings need to be attended to in order to customize KeyTerms for installation on your domain:

Deployment mode

Back to top

The first step in deploying KeyTerms is to determine whether to run in development or production mode. This choice slightly alters the behavior of Keyterms, affecting which ports are used by KeyTerms as well as the quantity and specificity of debugging output.

KeyTerms is configured to follow the Node.js convention of `NODE_ENV == 'dev' || 'prod'`. There are two main ways to specifty the KeyTerms in deployment mode:

1. before starting the server, from the command line or in a shell script `export NODE_ENV=prod`
2. while starting the server `NODE_ENV=prod npm start` from the application directory

Client connections and port configuration

Back to top

The KeyTerms server and the KeyTerms Client are separate applications that may optionally be served from the different servers. Because of this, and despite the fact that both may be hosted on the same machine, an explicit configuration setting exists to eliminate potential CORS issues.

Within `config.js`, there are two places to specify `allowedCorsDomains`, which is an array of URLs representing the allowable provenance of client connections. The two places where `allowedCorsDomains` should be set are within the configuration blocks for the *dev* and *prod* deployment modes.

Similarly, different ports may be opened by KeyTerms depending on whether you are running in production or development mode, and whether or not you are serving securely under SLL. The four places to specify ports in `config.js` are shown in the following snippet:

```
prod: {
        http: 5000,
        https: 5443,
        // NOTE: Make sure to use the correct protocol (http or https)
        allowedCorsDomains: ['http://localhost',
'http://keyterms.mycompany.com']
    },
    dev: {
        http: 4000,
        https: 4443,
        // NOTE: Make sure to use the correct protocol (http or https)
        allowedCorsDomains: ['http://localhost',
'http://keyterms.mycompany.com']
    },
```

The correct protocol, either http or https, must be specified to agree with whether or not you are hosting KeyTerms under SSL.

NLP services connection

Back to top

Nlp services is a Java web application. It provides various linguistic functions to KeyTerms, including normalization, tokenization, stemming and transliteration. The main configuration settings for connecting to NlpServices include:

- url - the URL where the service may be accessed
- useHTTPS - a boolean indicating whether NlpServices is deployed using SSL
- certs - a setting controlling which certs are to be used if useHTTPS is set to true

The relevant portion of config.js is:

```
// NLP Services configurations
config.nlp = {
    // base url for the NLP Services server
    url: 'http://keyterms.myorg.net:8080/NLPServices/TextServices',

    // if false, http is used rather than https
    useHTTPS: false,

    // these are the certs for NLP Services, only used if useHTTPS=true
    certs: config.server.SSLCerts           // by default, we re-use the same
certs the KeyTerms API is served with

    // uncomment this to include specific certs for NLP
    // certs: {
    //  key: __dirname+ '/certs/key.pem',
    //  cert: __dirname+ '/certs/cert.pem',
    //  ca: [__dirname+ '/certs/ca.pem']
    // }

};
```

MongoDB connection

Back to top

KeyTerms stores its data in MongoDB, and accordingly the KeyTerms config.js file must be updated to reflect the location and security settings for that database. The main configuration settings for MongoDB include:

- host - the name of the server hosting MongoDB
- port - the port on which the MongoDB connection is available
- db - the name of the database to be used in your installation (Note: this name can be altered if a fresh database is required. See creating a fresh database in an existing installation of KeyTerms for instructions.
- secured - a boolean indicating whether the Mongo database is username and password protected
- user - the username for KeyTerms to use when accessing the database, if the *secured* option is set to *true*
- pass - the password for KeyTerms to use when accessing the Mongo database, if the *secured* option is set to *true*

```
// Database (Mongo) configurations
config.db = {
    host: 'keyterms.myorg.net',
    port: 27017,
    db: 'KeyTerms',
    secured: false,     // set to true if Mongo instance is username/password
protected
    user: '',
    pass: ''
};
```

Note: If MongoDB is installed on a different machine than the one used to serve KeyTerms server, your MongoDB instance will need to be configured to accept connections from external machines. By default, mongo only accepts connections from `localhost (127.0.0.1)`. *For more information, see MongoDB Documentation (https://docs.mongodb.com/manual/).*

ElasticSearch connection

Back to top

ElasticSearch provides the index, search and query result ranking mechanisms of KeyTerms. The configuration settings and relevant portion of config.js for interfacing with ElasticSearch are:

- host - the name of the server hosting ElasticSearch
- port - the port on which the MongoDB connection is available
- protocol - the communications protocol for connecting to the ElasticSearch index, options {'http','https'}

```
// ElasticSearch configurations
config.elastic = {
    host: 'ktelastic.mycompany.org',
    port: 9200,
    protocol: 'http'
};
```

Note: If ElasticSearch is running on a separate machine, make sure to the ElasticSearch configuration to allow connections from external machines. By default, ElasticSearch is configured to only accept connections from `localhost(127.0.0.1)`. *For more information, see [ElasticSearch Documentation (https://www.elastic.co/guide/index.html)](https://www.elastic.co/guide/index.html).*

SSL security

Back to top

If deploying KeyTerms under SSL, there are several settings in the config.server section of config.js that need to be attended to.

- useHTTPS - a boolean indicating whether or not to use HTTPS. If this option is set to false, all other settings pertaining to SSL are ignored.
- https - two port settings for which https ports to use when deploying under prod or dev
- SSLCerts - a collection of settings for specifying the locations of the files for. Note that unless absolute paths are used to specify these file locations, KeyTerms server will search for them at the same directory level as the executing server configuration file
    - key - the KeyTerms server's private key
    - cert - the KeyTerms server's signed public-facing certificate
    - ca - a file listing one or more acceptable certificate authorities
- TLSOptions - a collection of settings available through NodeJS for setting the Transport Layer Security options (See the NodeJS documentation (https://nodejs.org/api/tls.html#tls_tls_ssl_concepts) for more detilas.
    - requestCert - a boolean indicating whether or not KeyTerms should request a certificate as a condition for accepting client connections
    - rejectUnauthorized - a boolean indicating whether or not KeyTerms should accept unauthorized client certificates, recommended setting is 'true'. Setting this to false is considered insecure.
    - secureProtocol - a string describing which TLS/SSL protocol NodeJS should use (See openSSL documentaiton (https://www.openssl.org/docs/man1.0.2/ssl/ssl.html#DEALING-WITH-PROTOCOL-METHODS) for available Strings
    - secureOptions - a list of optional TLS settings provided by NodeJS. The KeyTerms settings by default disallow various old or untrusted versions of TLS.
    - ciphers - a list of encryption algorithms accepted by the server
    - honorCipherOrder - A NodeJS SSL option controlling whether to use the server's preference order instead of the client's when choosing a cipher from the list of ciphers. The recommended setting for this is 'true'.

```
// Server configurations
config.server = {
    useHTTPS: false,
    cookieExpiration: 1000 * 60 * 60 * 24,      // one day in milliseconds
    prod: {
        http: 5000,
        https: 5443,
        // NOTE: Make sure to use the correct protocol (http or https)
        allowedCorsDomains: ['http://localhost',
'http://keyterms.mycompany.org']
    },
    dev: {
        http: 4000,
        https: 4443,
        // NOTE: Make sure to use the correct protocol (http or https)
        allowedCorsDomains: ['http://localhost',
'http://keyterms.mycompany.org']
    },
    SSLCerts: {
        key: __dirname+ '/certs/default.key',
        cert: __dirname+ '/certs/default.crt',
        ca: [__dirname+ '/certs/ca/ca.cert.pem']
    },
    TLSOptions: {
        requestCert: true,
        ~~**rejectUnauthorized: false,**~~
        secureProtocol: 'TLSv1_2_method',
        secureOptions: CONSTANTS.SSL_OP_NO_SSLv3 || CONSTANTS.SSL_OP_NO_SSLv2
||
            CONSTANTS.SSL_OP_NO_TLSv1 || CONSTANTS.SSL_OP_NO_TLSv1_1,
        ciphers: 'ECDHE-RSA-AES128-SHA256:AES128-GCM-
SHA256:HIGH:!RC4:!MD5:!aNULL:!EDH',
        honorCipherOrder: true
    }
};
```

Default Glossary

Back to top

The config.commonOrg section defines the properties of a common, or default, glossary that will be created upon install. All subsequent users will be added to this glossary by default upon user creation. The `init-cli` script, as mentioned below in setting up the database, will read from this config, so it is important to have it set before that script is run.

- name - The name of the common glossary.
- abbreviation - An abbreviation, or short name, for the common glossary.
- description - A description for the common glossary.

```
// Common (default) Organization configuration
config.commonOrg = {
    name: 'Common Glossary',
    abbreviation: 'commgloss',
    description: 'Glossary that all users share and belong to by default'
}
```

Setting up the database for first use

Back to top

The first time KeyTerms is installed, a script called `init-cli` needs to be run in order to create the MongoDB collections, establish an organization for the system administrators, and create the first KeyTerms administrator account. This script launches a command-line interface (CLI) that creates the common (default) glossary based on the default glossary config, and allows the user to create an initial Admin User account that is an admin of the common glossary.

The `init-cli` script should be run from within the KeyTerms server installation directory using the command `npm run init-cli`. This step is not required if a compatible database is still available from a previous KeyTerms installation. If a previous installation's database is to be re-used, then the name of the pre-existing database should be configured in the configuration portion of the KeyTerms server configuration file(#configuring-connections-to-the-mongodb).

Two other scripts are available, and are run similarly from within the server installation directory:

1. `npm run org-cli` - launches a command-line interface (CLI) allowing the user to create an Organization (for both convenience and for fresh installs)
2. `npm run admin-cli` - deprecated by the `init-cli` script, but can still be used. Launches a CLI allowing the user to create an Admin User account

Please note, it is impossible to access KeyTerms without an account, including the Admin Pages. Therefore these CLIs were created to allow the System Admin to create a user for themselves, to then access the admin pages at the /admin endpoint, for example (`localhost:4000/admin`).

Verifying the KeyTerms server installation

Back to top

There are a few ways to verify the successful installation of the KeyTerms Server.

1. **using curl**
2. **using a browser**

Testing the server using curl

Back to top

To star the server, run `npm start` from within $KTSERVER_BASE/app
To test your KeyTerms server run `curl localhost:4000`. Curl should return a JSON object reporting the system's status.

Testing the server using a browser

Back to top

Open a web browser and navigate to one of the endpoints, which can be constructed by appending the endpoint to your base server URL using the syntax *protocol://server:port/endpoint*, for example:

```
http://keyterms.mycompany.com:4000/docs
```

or

```
https://keyterms.mycompany.com:5000/api/schema
```

Server endpoints and paths

Back to top

All of the endpoints can be accessed programmatically with a GET operation.

- `/login` - simple login page.
- `/logout` - logs the current user out and destroys their session.
- `/admin` - *Sys Admins & Org Admins Only*. Separate GUI for user and org administration.
- `/upload` - *Sys Admins & Org QCs Only*. Separate GUI for uploading Excel spreadsheets and ingesting them into Organizations.
- `/api` - The root path of the KeyTerms API. Every endpoint on this path requires authentication, except:
    - `/api/status` - returns server data, including version
    - `/api/langcodes` - returns a JSON object of every supported language in KeyTerms
    - `/api/schema` - returns a JSON object representing the current `Entry` data model
    - `/api/api` - redirects to `/docs`. Exists for historical reasons
- `/docs` - returns a KeyTerms API reference
- `/whoami` - returns information about the currently logged in user

Installing the KeyTerms client

Back to top

The KeyTerms client has been tested using two methods for serving it: 1. using a NodeJS-based web server, and 2. using a Tomcat server. Descriptions of the client installation assume a directory structure similar to that described for the server installation. If both the client and the server were installed on the same machine, the resulting directory structure would be:

Sample directory structure for client and server on same machine:

```
$KEYTERMS_HOME
    serverKT-->$KTSERVER_BASE
    clientKT-->$KTCLIENT_BASE
    $KTSERVER_BASE
    $KTCLIENT_BASE
    config.js  (KeyTerms server configuration file)
```

Under this single-server setup, the client and the server are both installed under a single home directory for KeyTerms. Symbolic links within $KEYTERMS_HOME can be redirected to the current installation folders for the client and server.

Editing the client configuration file

Back to top

The KeyTerms client comes with a very simple configuration file located in $KTCLIENT_BASE/config.js. The settings include:

- **apiUrl** - This is required for TermBase to function properly. This need to be base url of the KeyTerms server instance this client web UI should interface with.
    - Example: `http://keyterms.mycompany.org:4000/`
- **mailto** - The email address users should be directed to when using the client's "Contact" tab

Deploy the KeyTerms client using a Node.js web server module

Back to top

Install and use a Node.js web server such as simple-server (https://www.npmjs.com/package/simple-server)

Deploy client using Apache Tomcat web server

Back to top

Create a keyterms application directory within the Apache Tomcat webapps directory. Copy the KeyTerms client static pages into that directory and restart the story.

PUT MORE DETAILS AND SAMPLE HERE

Verifying the client installation

Back to top

Using a web browser, navigate to the site where you expect your client to reside, constructing the URL using the syntax protocol://server:port/, for example:

```
http://keyterms.mycompany.com:8080
or
https://keyterms.mycompany.net:8446
```

It is best to test the client installation after launching the KeyTerms components

Launching the KeyTerms components

Back to top

All three of the main components need to be running for KeyTerms to function fully.

It is recommended to run all NodeJS processes as the a non-root user; for example, by creating a nodejs user as described in Unpacking the deployment files.

```
$ sudo su nodejs
            OR
$ sudo su tomcat
```

1. Serve NLPServices

```
$ sudo su tomcat
$ source $TOMCAT_HOME/bin/catalina.sh start
```

2. Start the KeyTerms server

```
$ sudo su nodejs
$ cd $KTSERVER_BASE/app
$ npm start
```

3. Serve the KeyTerms client

```
$ sudo su nodejs
$ cd $KTCLIENT_BASE/app
$ npm start
```

Running KeyTerms components as a service or a daemon

Back to top

Neither Node applications nor the Apache Tomcat web server will continue running once the ssh session that started them ends. Therefore, you will need to "daemonize" KeyTerms. There are a several ways to accomplish this. We recommend a a few here.

using nohup

Back to top

The nohup command can be used to run task that will not be interrupted when a terminal session is ended or interrupted. When run with an ampersand (&) at the end, nohup will run a task in as a background process that will not be interrupted when connections to the machine are terminated.

```
nohup node keyterms/app/app.js  &
```

Nohup by default redirects stdout into a file named nohup.out. If you have any concerns about the size of the nohup.out file, and your organization does not have rules requiring you to keep standard output as part of its logging requirements, the nohup.out file may be redirected to /dev/null, which is a dummy device used for disposing of unwanted output streams. Similarly, stderr may be redirected. In the following example, stderr is redirected to stdout, which is redirected to /dev/null.

```
nohup node keyterms/app/app.js > /dev/null 2>&1 &
```

If your organization requires either of these output streams to be kept for logging purposes, please consult a system administrator for advice on how to appropriately handle the redirection and storing of these outputs.

using specialized Node.js modules

Back to top

There are a few Node Modules that will run node applications continuously, including forever (https://www.npmjs.com/package/forever) or pm2 (https://www.npmjs.com/package/pm2).

```
$ sudo npm install -g forever
$ forever start keyterms/app/app.js
            OR
$ sudo npm install -g pm2
$ pm2 start keyterms/app/app.js
```

as a service, by installing a startup script that executes when your server boots

Back to top

create a startup script(link to sample script for ktserver and tomcatserver)
place script file into /etc/init.d
make sure script is owned by root
install service
using the service
service ktserver start
service ktserver status
service ktserver stop

Note: Sometimes a failed start will leave an invalid .pid file in the $KTSERVER_BASE directory. This file must be deleted before the service can be successfully re-started.

Updating KeyTerms

Updating KeyTerms versions

Back to top

-- Back up your previous server and client configuration files (or the whole directory).
-- Follow the installation procedures for a fresh install, using previous installation configuration files as a reference.
-- Decide whether to re-use previous database. Be sure to check new version's README to determine whether this is possible, since it is possible that major changes to the database structure might require a custom data migration.

Creating a fresh database in an existing installation

Back to top

If this is not a fresh install, but you would like to point KeyTerms to a fresh MongoDB database, perform the following steps:

1. shut down the KeyTerms server, if it is running
2. change the name of the database in the 'db' setting, withing the config.db section ofthe KeyTerms server's config.js file, as in the following example:

```
    // Database (Mongo) configurations
          config.db = {
              host: 'keyterms.myorg.org',
              port: 27017,
              db:  'KeyTerms_2018-01-08',
              secured: false,       // set to true if Mongo instance is
username/password protected
              user: '',
              pass: ''
          };
```

3. `npm run org-cli` - launches a CLI allowing the user to create an Organization (for both convenience and fresh installs)
4. `npm run admin-cli` - launches a CLI allowing the user to create an Admin User account (usually only used after a fresh install)
5. re-start the KeyTerms server

Installing Third-party Dependencies

In general, it is recommended that an experienced systems administrator or dev-ops engineer install third-party dependencies according to best practices recommended by each provider and according to your own organization's established practices. Some supplemental installation tips are provided here for your convenience. Please consult the relevant documentation if more detailed or custom guidance is required for your situation.

Installing Node.js

Back to top

Locate the latest Node.js 6.x version from 'https://nodejs.org/dist/latest-v6.x/'. Be sure to select an installer appropriate for your version of Linux.

- For detailed information on installing NodeJS from a file, consult the NodeJS documentation (https://github.com/nodejs/help/wiki/Installation).
- For detailed information on installing NodeJS via package manager, see the Node.js website (https://nodejs.org/en/download/package-manager/).

An example using the yum package manager:

```
yum install nodejs https://nodejs.org/en/download/releases/
```

Configuring the Node.js proxy server

Back to top

If you are behind a proxy server and need Node to access to the internet, for example for updating dependincies via 'npm install', npm will need to be configured to allow internet traffic:

```
npm config set proxy http://proxy.mycompany.com:8080
OR
npm config set https-proxy https://proxy.mycompany.com:443
```

Installing ElasticSearch

Back to top

For complete information on installing ElasticSearch, see the ElasticSearch documentation (https://www.elastic.co/guide/en/elasticsearch/reference/current/_installation.html).

Note: If ElasticSearch is running on a separate machine, make sure to edit its configuration to allow connections from external machines. By default, ElasticSearch is configured to only accept connections from `localhost(127.0.0.1)`

Installing MongoDB

Back to top

MongoDB installation instructions are very specific to the operating system and architecture used for deployment. Please consult the MongoDB website (https://resources.mongodb.com/getting-started-with-mongodb) for installation instructions appropriate to your situation. We provide some recommendations for your convenience:

- When installing via yum, *mongodb-org* is the package to install.
- Install version 3.4. Minor version updates of 3.4 such as 3.4.15 are acceptable, but KeyTerms has not been tested on later major versions.

Note: If your MongoDB installation is on a separate machine, make sure to edit the `mongod.conf` file to allow connections from external machines. By default, MongoDB only accepts connections from `localhost (127.0.0.1)`