Install and configure EclecticIQ Platform

Install and config guide for system administrators

Last generated: March 08, 2017

Last generated on Mar 8, 2017

Table of contents

How to install the platform via an RPM package

Summary: This step-by-step tutorial walks you through a fresh installation of the platform onto a virtual server via an RPM package.

Scope

This is a stripped-down, nitty-gritty version of the RPM installation and configuration guide. It walks you through the commands and the configuration settings you need to execute and apply to perform a fresh install of the EclecticIQ Platform on a target system.

Goal

The tutorial walks you through an RPM installation of the on an Amazon EC2 virtual server.
You can copy-paste the code examples and use them as they are or with minimal edits to accommodate your environment.

Audience

For the impatient system administrator.

Prerequisites

Credentials and host name

To correctly configure the system after installing the required products, ensure you have the following information available:

- DNS name of the host you are going to use to access the platform. Example: `platform.host`

- SSL certificate and key for the web server.
- EclecticIQ Platform login credentials.

SSH default login credentials for the VM OS	
user	`packer`
password	`Packer123!`

EclecticIQ Platform default login credentials	
user	`admin`
password	`EclecticIQ2015#`

Operating systems

Supported operating systems:

- **CentOS Linux 7 (1511)** (`https://lists.centos.org/pipermail/centos-announce/2015-december/021518.html`)
- **Red Hat Enterprise Linux 7** (`https://www.redhat.com/`)

> **SELinux** (`http://selinuxproject.org/`) is supported as per release 0.13.

Create the virtual server instance

- Go to **Amazon EC2** (`https://aws.amazon.com/ec2/`).
- Create an ***m4.xlarge*** (`https://aws.amazon.com/ec2/instance-types/`) instance.
- As the **base AMI** (`https://docs.aws.amazon.com/awsec2/latest/userguide/finding-an-ami.html`), use the latest CentOS Linux 7 x86_64 **HVM EBS** (`https://docs.aws.amazon.com/awsec2/latest/userguide/virtualization_types.html`).
- When the new virtual server is available, log in to the new instance with the `centos` user and the appropriate PPK file.

Configure the virtual server instance

- Replace `<hostname>` with the applicable host name for your environment:

```
$ sudo hostnamectl set-hostname <hostname>
```

- Add the host name to the *hosts* file:

```
$ sudo nano /etc/hosts
```

- Add the following test to the *hosts* file.
  Place these entries at the beginning of the loopback address section in the file.
  Replace `<hostname>` with the host name you just set:

```
127.0.0.1   <hostname>.localdomain   <hostname>
::1         <hostname>.localdomain   <hostname>
```

- Save the file and exit the text editor: on the keyboard, press **F3**, **ENTER**, and then **F2**.
- Check for new updates for the packages:

```
$ sudo yum check-update
```

- If any updates are available and ready to be installed, run the following command and apply all available updates:

```
$ sudo yum update
```

- Install *npm*, *unzip*, and *wget*.
  These utilities come in handy during the platform installation process:

```
$ sudo yum -y install npm unzip wget
```

- Make sure your RHEL or CentOS server includes **supervisor** `(http://supervisord.org/index.html)` and **systemd** `(https://freedesktop.org/wiki/software/systemd/)`, since they control and manage core platform processes, services, and task workers.
- To install Supervisor, run the following command(s):

```
$ sudo yum install supervisor
```

- Both process managers should be up and running before you proceed with the configuration of the platform and its components.
  By default, *systemd* should automatically start at system bootup. You can use it to check if *Supervisor* is enabled, too:

```
$ sudo systemctl status supervisord
```

- If necessary, enable *supervisord*, the daemon service:

```
$ sudo systemctl enable supervisord
```

- After enabling *supervisord*, start it:

```
$ sudo systemctl start supervisord
```

Add the repositories

- Add the following repositories to the system as root:

```
$ sudo wget https://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-
9.noarch.rpm && rpm -ivh epel-release-7-9.noarch.rpm

$ sudo wget https://centos7.iuscommunity.org/ius-release.rpm && rpm -ivh ius-
release.rpm

$ sudo wget https://download.postgresql.org/pub/repos/yum/9.5/redhat/rhel-7-
x86_64/pgdg-centos95-9.5-3.noarch.rpm && rpm -ivh pgdg-centos95-9.5-3.noarch.rpm
```

Install the core dependencies

EPEL

- Install **Extra Packages for Enterprise Linux (EPEL)** (https://fedoraproject.org/wiki/epel):

```
$ sudo yum -y install epel-release
```

IUS repository

- Install the **IUS** (https://ius.io/) repository for the IUS distribution of CentOS 7:

```
$ sudo yum -y install https://centos7.iuscommunity.org/ius-release.rpm
```

Java SE Development Kit

- Obtain **Oracle Java SE Development Kit 8** (http://www.oracle.com/technetwork/java/javase/downloads/), release 8u72 or later:

```
$ wget --no-cookies --header "Cookie: gpw_e24=http%3A%2F%2Fwww.oracle.com%2F;
oraclelicense=accept-securebackup-cookie" \
"http://download.oracle.com/otn-pub/java/jdk/8u74-b02/jdk-8u74-linux-x64.rpm"
```

- **Install** (http://tecadmin.net/install-java-8-on-centos-rhel-and-fedora/) Java SE Development Kit:

```
$ sudo yum install jdk-8u74-linux-x64.rpm
```

- Verify the JDK version installed on the target system:

```
$ java -version
```

Downloading Java on a Linux machine through the CLI may get you stranded on their license page.
If you experience this issue, the following resources provide workarounds to proceed with the product installation:

- ***jdk_download.sh* script on GitHub Gist** (https://gist.github.com/p7h/9741922)
- **JDK installation with `wget` example on GitHub Gist** (https://gist.github.com/scottvrosenthal/11187116)
- **JDK installation with `wget` examples on Stack Overflow** (https://stackoverflow.com/questions/10268583/downloading-java-jdk-on-linux-via-wget-is-shown-license-page-instead)

Install third-party components

Nginx

- Set up the **Nginx** (http://nginx.org/en/docs/) repository by creating the *nginx.repo* repository file:

```
$ sudo nano /etc/yum.repos.d/nginx.repo
```

- Paste the following lines to the *neo4j.repo* file:

```
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/7/x86_64/
gpgcheck=0
enabled=1
```

- Save the file and exit the text editor: on the keyboard, press **F3**, **ENTER**, and then **F2**.
- Install Nginx 1.8:

```
$ sudo yum install nginx-1.8*
```

- Verify that Nginx is correctly installed:

```
$ type nginx
```

PostgreSQL

- Download **PostgreSQL 9.5.3** (https://yum.postgresql.org/repopackages.php):

```
$ sudo yum -y install https://download.postgresql.org/pub/repos/yum/9.5/redhat/rhel-7-x86_64/pgdg-centos95-9.5-3.noarch.rpm
```

- Install **PostgreSQL 9.5.3** (https://yum.postgresql.org/repopackages.php):

```
$ sudo yum install -y postgresql95 postgresql95-server postgresql95-contrib
```

- Initialize the database:

```
$ /usr/pgsql-9.5/bin/postgresql95-setup initdb
```

- Enable the PostgreSQL service to start at system bootup:

```
$ sudo systemctl enable postgresql-9.5.service
```

- Start the PostgreSQL service:

```
$ sudo systemctl start postgresql-9.5.service
```

- Check if PostgreSQL is running:

```
$ sudo systemctl status postgresql-9.5.service
```

Redis

- Install **Redis 2.8.19-2.el7** (http://redis.io/download):

```
$ sudo yum install redis
```

ELK stack

Elasticsearch

- Download and install the public signing key:

```
$ rpm --import https://packages.elastic.co/GPG-KEY-elasticsearch
```

- Set up the Elasticsearch repository by creating the *elasticsearch.repo* repository file:

```
$ sudo nano /etc/yum.repos.d/elasticsearch.repo
```

- Paste the **following lines** (https://www.elastic.co/guide/en/elasticsearch/reference/current/setup-repositories.html) to the *elasticsearch.repo* file:

```
[elasticsearch-2.x]
name=Elasticsearch repository for 2.x packages
baseurl=https://packages.elastic.co/elasticsearch/2.x/centos
gpgcheck=1
gpgkey=https://packages.elastic.co/GPG-KEY-elasticsearch
enabled=1
```

- Save the file and exit the text editor: on the keyboard, press **F3**, **ENTER**, and then **F2**.
- Install **Elasticsearch 2.3** (https://www.elastic.co/guide/en/elasticsearch/reference/2.3/setup-repositories.html#_yum_dnf):

```
$ sudo yum install elasticsearch-2.3*
```

- Install elasticdump 2.4.2:

```
$ sudo npm install elasticdump@2.4.2 -g
```

Logstash

- Set up the Logstash repository by creating the *logstash.repo* repository file:

```
$ sudo nano /etc/yum.repos.d/logstash.repo
```

- Paste the **following lines** (https://www.elastic.co/guide/en/logstash/current/installing-logstash.html) to the *logstash.repo* file:

```
[logstash-2.3]
name=Logstash repository for 2.3.x packages
baseurl=https://packages.elastic.co/logstash/2.3/centos
gpgcheck=1
gpgkey=https://packages.elastic.co/GPG-KEY-elasticsearch
enabled=1
```

- Save the file and exit the text editor: on the keyboard, press **F3**, **ENTER**, and then **F2**.
- Install **Logstash** (https://www.elastic.co/downloads/logstash):

```
$ sudo yum install logstash
```

Kibana

- Set up the **Kibana** (http://nginx.org/en/docs/) repository by creating the *kibana.repo* repository file:

```
$ sudo nano /etc/yum.repos.d/kibana.repo
```

- Paste the following lines to the *kibana.repo* file:

```
[kibana-4.5]
name=Kibana repository for 4.5.x packages
baseurl=http://packages.elastic.co/kibana/4.5/centos
gpgcheck=1
gpgkey=http://packages.elastic.co/GPG-KEY-elasticsearch
enabled=1
```

- Install Kibana 4.5.1:

```
$ sudo yum install kibana-4.5.1
```

- Follow the steps outlined in the **official documentation** (https://www.elastic.co/downloads/kibana) and review the product **Readme file** (https://github.com/elastic/kibana/blob/4.5/readme.md).

Neo4j

- Set up the Neo4j repository by creating the *neo4j.repo* repository file:

```
$ sudo nano /etc/yum.repos.d/neo4j.repo
```

- Paste the **following lines** (http://optimalbi.com/blog/2016/03/15/how-to-install-neo4j-on-aws-linux/) to the *neo4j.repo* file:

```
[neo4j]
name=Neo4j Yum Repo
baseurl=http://yum.neo4j.org
enabled=1
gpgcheck=1
gpgkey=http://debian.neo4j.org/neotechnology.gpg.key
```

- Save the file and exit the text editor: on the keyboard, press **F3**, **ENTER**, and then **F2**.
- Install Neo4j 2.3.1 Community:

```
$ sudo yum install neo4j-2.3.1
```

ELK services

- Set up Elasticsearch to automatically start on boot:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable elasticsearch.service
```

- Start the Elasticsearch service:

```
$ sudo systemctl start elasticsearch.service
```

or:

```
$ sudo service elasticsearch start
```

Set up the EclecticIQ repository

- Set up the EclecticIQ repository by creating the *eclecticiq-platform.repo* repository file:

```
$ sudo nano /etc/yum.repos.d/eclecticiq-platform.repo
```

- Paste the following lines into the *eclecticiq-platform.repo* file.
  Replace `<username>` and `<password>` with the actual EclecticIQ credentials:

```
[eclecticiq-platform]
name=eclecticiq-platform
baseurl=https://downloads.eclecticiq.com/platform
gpgcheck=1
repo_gpgcheck=0
enabled=1
username=<username>
password=<password>
gpgkey=https://downloads.eclecticiq.com/platform/repodata/repomd.xml.key
```

- Save the file and exit the text editor: on the keyboard, press **F3**, **ENTER**, and then **F2**.
- The installation or upgrade procedure should normally take care of removing */etc/yum.repos.d/private-eclecticiq.repo*.
  In case it does not happen automatically, you can safely remove */etc/yum.repos.d/private-eclecticiq.repo* manually:”

```
$ sudo rm -fR /etc/yum.repos.d/private-eclecticiq.repo
```

- Force a refresh of the YUM cache to load the *eclecticiq-platform.repo* repository:

```
$ sudo yum makecache
```

Install the platform

- To install a specific platform release, run the following command(s):
  (The *x.x.x* part in the archive or in the RPM package file name is replaced in the actual files with the applicable platform release number, for example *eclecticiq-platform-1.14.0.zip* or *eclecticiq-platform-1.14.0.x86_64.rpm*. Make sure you replace *x.x.x* with the appropriate release number in the file name when you execute actions on these files.)

```
$ sudo yum -y install eclecticiq-platform-x.x.x

# For example, to install release 1.14.0:
$ sudo yum -y install eclecticiq-platform-1.14.0
```

- To install the latest platform release, run the following command(s):

```
$ sudo yum -y install eclecticiq-platform
```

- To install the latest platform release along with extra dependencies, run the following command(s):

```
$ sudo yum install -y eclecticiq*.rpm
```

Configure the platform

Third-party configuration files

After installing a dependency, look for the corresponding configuration file to edit it to reflect the target installation environment.

If you accept the default installation locations, you can find most configuration files related to the platform dependencies and third-party components under the following folders:

- */etc*
- */etc/<product_name>*
  For example: */etc/nginx/nginx.conf*

The platform ships with a set of reference configuration files for its dependencies. You can use them as guidelines or boilerplates to fine-tune your configuration files as appropriate.
In most cases, you can replace the configuration file created during the third-party component installation with the corresponding one available in */opt/eclecticiq/etc-extras/*, and you can apply minimal edits to tailor it to your target environment.

Review these files and edit them, where necessary. Refer to the component official documentation for specific details.
The reference configuration files are stored in the following folder and inside its sub-folders:

```
$ cd /opt/eclecticiq/etc-extras/
```

This is an overview of the reference configuration files for platform dependencies that you can use as boilerplates to configure third-party components:

```
# Kibana config file
/opt/eclecticiq/etc-extras/kibana.yml

# Redis message broker config file
/opt/eclecticiq/etc-extras/redis.conf

# Elasticsearch config files
/opt/eclecticiq/etc-extras/elasticsearch/elasticsearch.yml
/opt/eclecticiq/etc-extras/elasticsearch/logging.yml

# Neo4j config files
/opt/eclecticiq/etc-extras/neo4j/neo4j.properties
/opt/eclecticiq/etc-extras/neo4j/neo4j-server.properties
/opt/eclecticiq/etc-extras/neo4j/neo4j-wrapper.conf

# Nginx web server config file
/opt/eclecticiq/etc-extras/nginx/nginx.conf

# Postfix email server config file
/opt/eclecticiq/etc-extras/postfix/main.cf

# Kibana and Neo4j ini files
/opt/eclecticiq/etc-extras/supervisord/kibana.ini
/opt/eclecticiq/etc-extras/supervisord/neo4j-console.ini
```

Create the database

- Connect to PostgreSQL with root privileges, and access it with the `postgres` user name:

```
$ su - postgres
$ psql
```

- In PostgreSQL, do the following:
  - **Create** (http://www.postgresql.org/docs/current/static/tutorial-createdb.html) a new database (in the example: *platform*).
  - Create a new user, and assign them a password (in the example: *test*/*test*, respectively).
  - Grant the new user full privileges on the newly created database.

```
CREATE DATABASE platform;
CREATE USER test WITH PASSWORD 'test';
GRANT ALL PRIVILEGES ON DATABASE platform to test;
```

Add the database to the platform settings

- In the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* platform configuration file, update the `SQLALCHEMY_DATABASE_URI` parameter with the designated database URI for your environment:

```
SQLALCHEMY_DATABASE_URI = "postgresql://<username>:<password>@<host>:<port>/<database-name>"
```

`username`	Replace this placeholder with the user name of the user that can access the database. Example: `test`.
`password`	Replace this placeholder with the corresponding user password. Example: `test`.
`host`	Replace this placeholder with the host name identifying the location where the database is hosted. Example: `127.0.0.1`.
`port`	The database access port. Default value: `5432`.
`database-name`	The assigned name to identify the database. Example: `platform`.

Set the database authentication method

- Open the *pg_hba.conf* configuration file in a text editor:

```
$ sudo nano /var/lib/pgsql/9.5/data/pg_hba.conf
```

- Set the authentication method to `trust` for the `local` type, and to `md5` for the `host` type to request an MD5-encrypted password:

```
# TYPE    DATABASE    USER       ADDRESS    METHOD
  local   all         test                  trust
  host    all         all        samenet    md5
```

Restart the database service

- Restart the PostgreSQL service:

```
$ sudo systemctl restart postgresql-9.5.service
```

or:

```
$ sudo service postgresql-9.5 restart
```

- Verify that user and database are successfully created:

```
$ psql -U test -d platform -W
```

Configure Nginx

To configure Nginx, do the following:

- Go to */etc/nginx*.
  You may need to access the resource by prefixing `sudo su`, so that you can obtain root privileges.
- Back up */etc/nginx/nginx.conf*, and replace it with the reference *nginx.conf* file shipped with the platform:

```
$ sudo su
$ cp /opt/eclecticiq/etc-extras/nginx/nginx.conf /etc/nginx/nginx.conf
```

- Edit */etc/nginx/nginx.conf* to reflect your actual configuration:

```
$ sudo su
$ sudo nano /etc/nginx/nginx.conf
```

Set Nginx user and group

Check the following parameters and their settings, since they affect interoperability between the platform and Nginx:

- The `user` should be set to `nginx nginx` for user name and group name, respectively:

```
user    nginx nginx;
```

Check access log format

Verify the Nginx **access log** (https://www.nginx.com/resources/admin-guide/logging-and-monitoring/) format to make sure Nginx can recognize and consume the *expected format for web server logs*.

- The `log_format` directive holds the configuration parameters for the **log format** (https://nginx.org/en/docs/http/ngx_http_log_module.html#log_format).
  As per Nginx 1.8, the following example shows the default JSON format for web server logs:

```
log_format json '{'
                    '"remote_addr": "$remote_addr",'
                    '"remote_user": "$remote_user",'
                    '"time_local": "$time_local",'
                    '"request": "$request",'
                    '"status": $status,'
                    '"body_bytes_sent": $body_bytes_sent,'
                    '"http_referer": "$http_referer",'
                    '"http_user_agent": "$http_user_agent",'
                    '"http_x_forwarded_for": "$http_x_forwarded_for"'
                    '}';
```

- Make sure the **access_log** (https://nginx.org/en/docs/http/ngx_http_log_module.html#access_log) `format` value matches the corresponding value specified in `log_format <format_name>`.
  The default Nginx log format for EclecticIQ Platform is `json`:

```
access_log /var/log/nginx/access.log json;
```

Set Nginx to load the platform configuration

- To enable Nginx to pick up and start the platform configuration, copy *platform.conf*
  - from */opt/eclecticiq/etc/nginx/conf.d/platform.conf*
  - to */etc/nginx/conf.d*

```
$ cp /opt/eclecticiq/etc/nginx/conf.d/platform.conf /etc/nginx/conf.d/platform.conf
```

- Open *platform.conf* in a text editor:

```
$ sudo nano /etc/nginx/conf.d/platform.conf
```

- Look for the `server_name` property.
- Replace `server_name <default_server_name>;` with the appropriate platform host name for your environment:

```
// Default server name value:
server_name <default_server_name>;

// Change it to your platform host name:
server_name <platform_server_name>;
```

- Make sure that the SSL certificate paths point to the correct locations where your certificates are stored. Example:

```
ssl_certificate      /etc/pki/tls/certs/local.iw.crt;
ssl_certificate_key  /etc/pki/tls/private/local.iw.key;
```

Load Nginx at bootup

- Enable the Nginx service to start at system bootup:

```
$ sudo systemctl enable nginx
```

Reload the Nginx configuration

- **Reload** (http://nginx.org/en/docs/beginners_guide.html#control) the Nginx configuration and start the new worker process with a new/updated configuration:

```
$ sudo service nginx reload
```

Start Nginx

- Start the Nginx web server:

```
$ sudo systemctl start nginx
```

or:

```
$ sudo service nginx start
```

Configure Redis

To configure Redis, do the following:

- Go to */etc*.
  You may need to access the resource by prefixing `sudo su`, so that you can obtain root privileges.
- Back up */etc/redis.conf*, and replace it with the reference *redis.conf* file shipped with the platform:

```
$ sudo su
$ cp /opt/eclecticiq/etc-extras/redis.conf /etc/redis.conf
```

For reference, look up a copy of the **self-documented file** (`https://raw.githubusercontent.com/antirez/redis/2.8/redis.conf`), and the **Redis configuration documentation** (`https://redis.io/topics/config`).

Check the Redis configuration

- Edit */etc/redis.conf* to reflect your actual configuration.
- Check the following parameters and their settings, since they affect interoperability between the platform and Redis:
  - `port 6379`: this is the default port for Redis.
  - `dbfilename dump.rdb`: the default file name of the database Redis saves snapshots to.
  - `dir /<some_dir>/<redis_snapshot>`: sets the location where Redis stores the snapshot database.

Check permissions

Make sure the target directory to save application data to exists before setting it in the configuration file, and that it is accessible to read and write data. If necessary, do the following:

- Create the target directory:

```
$ sudo mkdir /<some_dir>/<target_dir>
```

- Make sure the correct user owns the target directory to access it in read and write modes.
  For example, if you set the data storage location to `/media/redis`, the `redis` user should own it:

```
$ sudo chown -R redis:redis /media/redis
```

Check the platform settings

- Open */opt/eclecticiq/etc/eclecticiq/platform_settings.py*, and verify that `REDIS_URL` points to the correct location in your environment:

```
REDIS_URL="redis://<redis.host>:<redis.port>"
```

- To verify that Redis is correctly installed, do the following:

```
$ type redis-server

$ type redis-cli
```

Start Redis

- To start Redis, run the following command(s):

```
$ sudo service redis start
```

- To check if Redis is running, run the following command(s):

```
$ sudo service redis status
```

- To verify that Redis is working properly, run the following command(s):

```
# Ping Redis to ask how it is doing
$ redis-cli ping

# Redis responds to confirm everything is ok
PONG
```

Configure Elasticsearch

To configure Elasticsearch, do the following:

- Go to */etc/elasticsearch*.
  You may need to access the resource by prefixing `sudo su`, so that you can obtain root privileges.
- Back up */etc/elasticsearch/elasticsearch.yml*, and replace it with the reference *elasticsearch.yml* file shipped with the platform:

```
$ sudo su
$ cp /opt/eclecticiq/etc-extras/elasticsearch/elasticsearch.yml
/etc/elasticsearch/elasticsearch.yml
```

- Edit */etc/elasticsearch/elasticsearch.yml* to reflect your actual configuration:

```
$ sudo su
$ nano /etc/elasticsearch/elasticsearch.yml
```

Check the following parameters and their settings, since they affect interoperability between the platform and Elasticsearch:

- Set `path.data` to the location where Elasticsearch stores its data.

Check permissions

Make sure the target directory to save application data to exists before setting it in the configuration file, and that it is accessible to read and write data. If necessary, do the following:

- Create the target directory:

```
$ sudo mkdir /<some_dir>/<target_dir>
```

- Make sure the correct user owns the target directory to access it in read and write modes.
  For example, if you set the data storage location to `/media/elasticsearch`, the `elasticsearch` user should own it:

```
$ sudo chown -R elasticsearch:elasticsearch /media/elasticsearch
```

Disable dynamic scripting

- You should disable **dynamic scripting** `(https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-scripting.html#enable-dynamic-scripting)` by either removing the `script.inline` property from the configuration file, or by setting it to `false`:

```
script.inline: false
script.indexed: true
```

Reference *elasticsearch.yml* file:

```
config:

  cluster.name: intel

  discovery.zen.ping.multicast: false

  index.number_of_replicas: 0

  index.number_of_shards: 1

  node.local: true

  path:

    data: /<path_to>/<elasticsearch_index_datastore>

  script.indexed: true

  script.inline: false
```

Check the Elasticsearch URL

- Open */opt/eclecticiq/etc/eclecticiq/platform_settings.py*, and verify that `SEARCH_URL` points to the correct location corresponding to the Elasticsearch instance:

```
SEARCH_URL="http://<elasticsearch.host>:<elasticsearch.port>"
```

Install the required plugins

- Install the following plugins:
  - *Optional* — **mobz/elasticsearch-head** `(https://github.com/mobz/elasticsearch-head)`
  - *Optional* — **codelibs/elasticsearch-reindexing** `(https://github.com/codelibs/elasticsearch-reindexing)`
  - **Required — delete-by-query** `(https://www.elastic.co/guide/en/elasticsearch/plugins/2.3/plugins-delete-by-query.html)`
    To install *delete-by-query*, run the following command(s):

```
$ /usr/share/elasticsearch/bin/plugin install delete-by-query
```

Configure Logstash

Logstash configuration files are copied automatically to the correct location during the post-install step.
To verify that the configuration files exist in the target destination, run the following command(s):

```
$ ls -l /opt/eclecticiq/etc/logstash/conf.d/
```

These are the Logstash configuration files related to the platform:

```
filters.conf
input.conf
neo4j-batching.conf
opentaxii.conf
output.conf
platform-api.conf
platform-ui.conf
```

The following files control the **Logstash processing pipeline** (https://www.elastic.co/guide/en/logstash/current/pipeline.html):

```
input.conf
filters.conf
output.conf
```

For further details, see the how-to article on addressing logging issues in Kibana and Logstash configurations.

Check platform-api.conf

- To check the *platform-api.conf* configuration file, run the following command(s):

```
$ nano /opt/eclecticiq/etc/logstash/conf.d/platform-api.conf
```

The *platform-api.conf* file holds the paths to core platform logs recording events related to ingestion, graph, search, web server, tasks workers and queues.
Verify that the paths to these log files are correct; edit them, if necessary.

```
input {
  file {
    path => ["/opt/eclecticiq/logs/platform-api.log", "/var/log/nginx/eclecticiq-
platform-api-nginx-access.log", "/opt/eclecticiq/logs/intel-ingestion.log",
"/opt/eclecticiq/logs/graph-ingestion.log", "/opt/eclecticiq/logs/search-
ingestion.log"]
    codec => "json"
    tags => "platform-api"
  }
  file {
    path => ["/opt/eclecticiq/logs/intel-ingestion.log", "/opt/eclecticiq/logs/graph-
ingestion.log", "/opt/eclecticiq/logs/search-ingestion.log"]
    codec => "json"
    tags => "ingestion"
  }
  file {
    path => ["/var/log/nginx/eclecticiq-platform-api-nginx-errors.log"]
    type => "nginx-error"
    tags => "platform-api"
  }
  file {
    path => ["/opt/eclecticiq/logs/task-worker-*.log", "/opt/eclecticiq/logs/task-
beat.log", "/opt/eclecticiq/logs/task-queue-analysis-listener.log",
"/opt/eclecticiq/logs/task-queue-enrichment-listener.log"]
    codec => "json"
    tags => "task-workers"
  }
}
```

platform-ui.conf, *opentaxii.conf*, and *neo4j-batching.conf* are similar to *platform-api.conf*: these files hold the paths to logs recording events related to the web-based user interface, to the TAXII transport type for STIX data, and to graph processing, respectively.
Verify that the paths to these log files are correct; edit them, if necessary.

Check elasticsearch.yml

You check this file to make sure that the `cluster.name` property is correctly set.
You may need to access the resource by prefixing `sudo su`, so that you can obtain root privileges.

- To open the *elasticsearch.yml* configuration file, run the following command(s):

```
$ sudo su
$ nano /etc/elasticsearch/elasticsearch.yml
```

The `config.cluster.name` property value should be `intel`:

```
config:

  cluster.name: intel

  discovery.zen.ping.multicast: false

  index.number_of_replicas: 0

  index.number_of_shards: 1

  node.local: true

  path:

    data: /<path_to>/<elasticsearch_index_datastore>

  script.indexed: true

  script.inline: false
```

Check the event pipeline

`input.conf`, `filter.conf`, and `output.conf` are the configuration files that control the Logstash **event processing pipeline** `(https://www.elastic.co/guide/en/logstash/current/pipeline.html)`:

- `input.conf` controls data ingestion into Logstash.
- `filter.conf` controls data manipulation like filtering, parsing and tranformations.
- `output.conf` controls data output, for example, by sending the processed data on to Elasticsearch.

Check input.conf

- To check the *input.conf* configuration file, run the following command(s):

```
$ nano /opt/eclecticiq/etc/logstash/conf.d/input.conf
```

input.conf holds the input configuration for **Elasticsearch Curator** `(https://www.elastic.co/guide/en/elasticsearch/client/curator/current/index.html)`, the Elasticsearch index manager.
You should not need to edit this file.

```
input {
  tcp {
    codec => json
    type => "curator"
    port => "28778"
  }
}
```

Check filters.conf

- To check the *filters.conf* configuration file, run the following command(s):

```
$ nano /opt/eclecticiq/etc/logstash/conf.d/filters.conf
```

Logstash filters (https://www.elastic.co/guide/en/logstash/current/config-examples.html) are like a Swiss-army knife to slice and dice your data using an array of **filter plugins** (https://www.elastic.co/guide/en/logstash/current/filter-plugins.html).
filters.conf defines filters as needed to select specific indices or data types based on the specified patterns. You can edit this file to match your requirements.
This *filters.conf* file is just an example that you can customize as needed:

```
filter {

  if [type] == "syslog" {
    grok {
      match => { "message" => "%{SYSLOG5424PRI}%{NONNEGINT:ver} +(?:%
{TIMESTAMP_ISO8601:ts}|-) +(?:%{HOSTNAME:containerid}|-) +(?:%
{NOTSPACE:containername}|-) +(?:%{NOTSPACE:proc}|-) +(?:%{WORD:msgid}|-) +(?:%
{SYSLOG5424SD:sd}|-|) +%{GREEDYDATA:msg}" }
    }
    syslog_pri { }
    date {
      match => [ "syslog_timestamp", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss" ]
    }
    if !("_grokparsefailure" in [tags]) {
      mutate {
        replace => [ "@source_host", "%{syslog_hostname}" ]
        replace => [ "@message", "%{syslog_message}" ]
      }
    }
    mutate {
      remove_field => [ "syslog_hostname", "syslog_message", "syslog_timestamp" ]
    }
  }

}
```

Check output.conf

- To open the *output.conf* configuration file, run the following command(s):

```
$ nano /opt/eclecticiq/etc/logstash/conf.d/output.conf
```

output.conf routes Logstash data downstream in the system toolchain. **Output plugins** (https://www.elastic.co/guide/en/logstash/current/output-plugins.html) send data in a predefined format to a specific destination.
Codec plugins (https://www.elastic.co/guide/en/logstash/current/codec-plugins.html) read and represent data in specific formats.

Within the platform environment, you can use the reference *output.conf* provided:

- It outputs Logstash data to the default Elasticsearch instance configured for the platform.
  By default, the Elasticsearch host is `localhost`, and the default Elasticsearch port is `9200`.
- `stdout` **encodes the data** `(https://www.elastic.co/guide/en/logstash/current/plugins-outputs-stdout.htmlencodes)` before outputting it to the shell standard output. You may edit it as needed.

```
output {
  elasticsearch { hosts => ["localhost:9200"] }
  stdout { codec => rubydebug }
}
```

Test the configuration

- To test the Logstash configuration, run the following command(s):

```
$ /opt/logstash/bin/logstash --configtest --config /etc/logstash/conf.d/
```

- If the configuration test result is positive, the response reads:

```
Configuration OK
```

Start Logstash

- Enable the Logstash service to start at system bootup:

```
$ sudo systemctl enable logstash
```

- To start Logstash, run the following command(s):

```
$ sudo systemctl start logstash
```

Configure Kibana

- To configure Kibana, simply replace */opt/kibana/config/kibana.yml* with the provided */opt/eclecticiq/etc-extras/kibana.yml*.
  The default settings in the */opt/eclecticiq/etc-extras/kibana.yml* configuration file should work in your environment without requiring any changes.

```
$ cp /opt/eclecticiq/etc-extras/kibana.yml /opt/kibana/config/kibana.yml
```

Kibana uses an index in Elasticsearch to store saved searches, visualizations, and dashboards.
The default property/value pair defining the index in the *kibana.yml* configuration file is `kibana_index: -kibana`.

The default parameters and values should not need any editing:

```
port: 5601
host: "0.0.0.0"
elasticsearch_url: "http://127.0.0.1:9200"
elasticsearch_preserve_host: true
kibana_index: "-kibana"
server.basePath: "/api/kibana"
default_app_id: "discover"
request_timeout: 300000
shard_timeout: 0
verify_ssl: false
bundled_plugin_ids:
 - plugins/dashboard/index
 - plugins/discover/index
 - plugins/doc/index
 - plugins/kibana/index
 - plugins/markdown_vis/index
 - plugins/metric_vis/index
 - plugins/settings/index
 - plugins/table_vis/index
 - plugins/vis_types/index
 - plugins/visualize/index
```

Check permissions

Kibana 4.5.x has an **issue** `(https://github.com/elastic/kibana/issues/6730)` that causes root owned files in */opt/kibana/optimize/*. This can prevent Kibana from running.
It should be fixed in version 4.6.0.

To work around it, you can change permissions so that the `kibana` user owns or has read/write access to the */opt/kibana/optimize/* directory.
To do so, run the following command(s):

```
$ sudo chown -Rvf kibana:kibana /opt/kibana/optimize/*
```

Check the Supervisor configuration

- Verify that the */etc/supervisord.conf* file includes a `[inet_http_server]` **section** `(http://supervisord.org/configuration.html#inet-http-server-section-settings)` with the properties enabling the `supervisord` daemon to respond to requests from the platform.
  To enable `inet_http_server`, */etc/supervisord.conf* should contain at least the following properties under `[inet_http_server]`:

```
[inet_http_server]
port=127.0.0.1:9001
```

Check the platform configuration

- Verify that the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* file includes the following line:

```
SUPERVISORD_HOSTS = '127.0.0.1:9001
```

Restart Supervisor

- After changing the Supervisor configuration, restart Supervisor:

```
$ sudo service supervisord restart
```

or:

```
$ systemctl restart supervisord
```

When you modify or update the Supervisor configuration, you need to run `$ sudo supervisorctl reload` to update it and to reload the up-to-date configuration into the platform.

- Reload the Supervisor configuration to restart all tasks and processes:

```
$ sudo supervisorctl reload
```

Start Kibana

- You can now start Kibana:

```
$ sudo service kibana start
```

or:

```
$ sudo systemctl start kibana
```

- To verify that Kibana is running, run the following command(s):

```
$ sudo service kibana status
```

or:

```
$ sudo systemctl status kibana
```

Configure Neo4j

To configure Neo4j, do the following:

- Go to */opt/eclecticiq/etc-extras/neo4j*.
- Copy the Neo4j configuration files shipped with the platform to */etc/neo4j*, so that they replace the default configuration files created during the Neo4j installation:

```
$ cp /opt/eclecticiq/etc-extras/neo4j/* /etc/neo4j/
```

The action should copy the following Neo4j configuration files to the destination directory:

```
neo4j-server.properties
neo4j-wrapper.conf
neo4j.properties
```

Check Neo4j connectivity

Check the main connectivity properties to verify that Neo4j can communicate with other components.

Check neo4j-server.properties

- Open the *neo4j-server.properties* configuration file:

```
$ nano /etc/neo4j/neo4j-server.properties
```

- Make sure that the following properties are set to the corresponding values:

```
org.neo4j.server.webserver.address=0.0.0.0
org.neo4j.server.webserver.port=7474
org.neo4j.server.database.location=/<some_dir>/graph.db
dbms.security.auth_enabled=False
```

`graph.db` is the default name of the graph database file.
Make sure the directory where it is stored exists before setting it in this configuration file, and that Neo4j can read and write data to it.

Inspect the amount of allocated memory: Neo4j purrs when you give it plenty of memory to work with.

- Depending on your system resources, allocate at least 4 GB to Neo4j.
- Never set it below 1 GB.

To set the minimum and maximum memory values for Neo4j, do the following:

Check neo4j-wrapper.conf

- Open the *neo4j-wrapper.conf* configuration file:

```
$ nano /etc/neo4j/neo4j-wrapper.conf
```

- Make sure that the following properties are set to the corresponding values:

```
wrapper.java.initmemory=1024
wrapper.java.maxmemory=4096
```

- The recommended value for `wrapper.java.initmemory` is at least 4096 MB or more, if possible. The minimum value should never be lower than 1024 MB.
- The recommended value for `wrapper.java.maxmemory` is at least 4096 MB or more, if possible. This value should never be lower than `wrapper.java.initmemory`.

Check neo4j.properties

Set the size of the page cache to `5G`:

- Open the *neo4j.properties* property file:

```
$ nano /etc/neo4j/neo4j.properties
```

- Make sure the `dbms.pagecache.memory` property is set to `5G`:

```
dbms.pagecache.memory=5G
```

If this parameter is not configured, Neo4j defaults to using 50% of the available system RAM for page caching.

Check permissions

Make sure the target directory to save application data to exists before setting it in the configuration file, and that it is accessible to read and write data. If necessary, do the following:

- Create the target directory:

```
$ sudo mkdir /<some_dir>/<target_dir>
```

- Make sure the correct user owns the target directory to access it in read and write modes. For example, if you set the data storage location to `/media/neo4j`, the `neo4j` user should own it:

```
$ sudo chown -R neo4j:neo4j /media/neo4j
```

Check the Neo4j URL

- Open */opt/eclecticiq/etc/eclecticiq/platform_settings.py*, and verify that `NEO4J_URL` points to the correct Neo4j instance in your environment:

```
NEO4J_URL="http://<Neo4j.host>:<Neo4j.port>"
```

Update the platform settings

- Review the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* platform settings file to make sure it reflects your environment.
  Review paths, URLs, and port numbers to verify that they match the corresponding settings in the configuration files for Elasticsearch, Kibana, Neo4j, Redis, and so on.

```
PLATFORM_MANIFESTS_DIR = '/opt/eclecticiq/manifests'
COMPONENT_SHARED_SECRET = "<component_secret_key_value>"
SECRET_KEY = "<secret_key_value>"
PROXY_URL_FILE_PATH = '/opt/eclecticiq/etc/eclecticiq/proxy_url'
SQLALCHEMY_DATABASE_URI = "postgresql://<username>:<password>@localhost:5432/platform"
SQLALCHEMY_POOL_SIZE = 20
SQLALCHEMY_MAX_OVERFLOW = 20
DISCOVERY_WORKSPACE_SIZE_LIMIT = 500
DISCOVERY_SEARCH_LIMIT = 1000
DISCOVERY_RESULTS_LIMIT = 500
INGESTION_QUEUE_NAME = 'queue:ingestion:inbound'
ENRICHMENT_QUEUE_NAME = 'queue:enrichment:inbound'
GRAPH_QUEUE_NAME = 'queue:graph:inbound'
GRAPH_INGESTION_SPOOL_DIRECTORY = '/opt/eclecticiq/tmp/gi-storage'
SEARCH_QUEUE_NAME = 'queue:search:inbound'
REDIS_URL = 'redis://127.0.0.1:6379/'
KIBANA_VERSION = '4.5.1'
KIBANA_URL = 'http://127.0.0.1:5601'
NEO4J_URL = 'http://127.0.0.1:7474'
NEO4J_BATCHING_URL = 'http://127.0.0.1:4008'
NEO4J_DEBUG = False
SEARCH_URL = 'http://127.0.0.1:9200'
OPENRESOLVE_URL = "http://api.openresolve.com/{}/{}"
RIPE_STAT_URL = "https://stat.ripe.net/data/{}/data.json?resource={}"
VIRUSTOTAL_API_SECRET = '<virustotal_api_secret_value>'
VIRUSTOTAL_API_URL = "https://www.virustotal.com/vtapi/v2/{}"
CELERY_QUEUES = [{"name": "providers", "routing_key": "eiq.providers.#"}, {"name":
"analysis", "routing_key": "eiq.analysis.#"}, {"name": "integrations", "routing_key":
"eiq.integrations.#"}, {"name": "enrichers", "routing_key": "eiq.enrichers.#"},
{"name": "utilities", "routing_key": "eiq.utilities.#"}, {"name":
"priority_enrichers", "routing_key": "eiq.priority.enrichers.#"}, {"name":
"priority_providers", "routing_key": "eiq.priority.providers.#"}, {"name":
"priority_utilities", "routing_key": "eiq.priority.utilities.#", {"name":
"reindexing", "routing_key": "eiq.reindexing.#"}]
```

- Review the */opt/eclecticiq/etc/eclecticiq/opentaxii.yml* OpenTAXII settings to make sure they reflect your environment.
  The only value you need to edit here should be `domain`: assign it the actual domain name of the system hosting the platform.

```
auth_api:
    class: eiq.opentaxii.auth.PlatformAuthAPI
    parameters: null
domain: <platform_host_name>
hooks: null
logging:
    opentaxii: debug
    root: debug
persistence_api:
    class: eiq.opentaxii.persistence.PlatformPersistenceAPI
    parameters: null
```

Bootstrap the platform

Before bootstrapping the platform and running the fixtures, make sure all processes and tasks are up and running.

- After changing the Supervisor configuration, restart Supervisor:

```
$ sudo service supervisord restart
```

or:

```
$ systemctl restart supervisord
```

- Reload the Supervisor configuration to restart all tasks and processes:

```
$ sudo supervisorctl reload
```

- Check Supervisor task statuses by running the following command(s):

```
$ supervisorctl status
```

The response should be similar to this example:

```
graph-ingestion                  RUNNING   pid 3443,  uptime 1:10:56
intel-ingestion:0                RUNNING   pid 3323,  uptime 1:11:19
intel-ingestion:1                RUNNING   pid 3336,  uptime 1:11:14
intel-ingestion:2                RUNNING   pid 3363,  uptime 1:11:09
intel-ingestion:3                RUNNING   pid 3372,  uptime 1:11:04
kibana                           RUNNING   pid 13837, uptime 9 days, 0:38:17
neo4j-batching                   RUNNING   pid 3590,  uptime 1:10:45
opentaxii                        RUNNING   pid 3662,  uptime 1:10:39
platform-api                     RUNNING   pid 2999,  uptime 1:12:47
search-ingestion                 RUNNING   pid 3513,  uptime 1:10:49
task:beat                        RUNNING   pid 3102,  uptime 1:12:39
task:enrichers                   RUNNING   pid 3110,  uptime 1:12:26
task:integrations                RUNNING   pid 3135,  uptime 1:12:13
task:providers                   RUNNING   pid 3204,  uptime 1:12:01
task:reindexing                  RUNNING   pid 3223,  uptime 1:11:48
task:utilities                   RUNNING   pid 3242,  uptime 1:11:35
```

Set up the database

- Create the database schema by running the following command(s):

```
$ /opt/eclecticiq/migrations/create-db.sh
```

- Generate the default platform fixtures by running the following command(s):

```
$ /opt/eclecticiq/migrations/load-fixtures.sh
```

Bootstrap Elasticsearch

Since we made some changes to the *elasticsearch.yml* configuration file in the previous steps, it is a good idea to stop and restart Elasticsearch to make sure it picks up the latest configuration settings.

- Stop Elasticsearch:

```
# Stop Elasticsearch
$ sudo service elasticsearch stop

# Response
Stopping elasticsearch (via systemctl):    [  OK  ]
```

- Start Elasticsearch:

```
# Start Elasticsearch
$ sudo service elasticsearch start

# Response
Starting elasticsearch (via systemctl):    [  OK  ]
```

- Create the Elasticsearch index by running the following command(s):

```
$ /opt/eclecticiq/platform/api/bin/manage synchronize_elasticsearch
```

Bootstrap Neo4j

Before proceeding to bootstrap Neo4j, verify that the following conditions are met:

- Make sure the Neo4j settings in the *platform_settings.py* configuration file are correct, based on your system environment.
  Typical/Default values are:

```
NEO4J_URL: 'http://localhost:7474'
NEO4J_BATCHING_URL = 'http://127.0.0.1:4008'
NEO4J_DEBUG: False
```

`neo4j-batching`	Neo4j graph database batch processing application. It lives on the same server hosting the Neo4j database. It prepares data for ingestion into the Neo4j database

Neo4j should be up and running:

- Check if Neo4j is running by making a cURL call to the URL defined in `NEO4J_URL` in the in the *platform_settings.py* configuration file.
  By default, it is `http://localhost:7474`.
- If Neo4j is not running, restart the service by running the following command(s):

```
$ sudo service neo4j start
```

or:

```
$ sudo supervisorctl start neo4j
```

- Verify that Neo4j is running:

```
$ sudo service neo4j status
```

Create the graph schema

- Before creating the graph schema, stop the `graph-ingestion` and any running `intel-ingestion` tasks:

```
$ sudo supervisorctl stop graph-ingestion
$ sudo supervisorctl stop intel-ingestion:*
```

- Then, run the following command to create the graph schema and to apply the necessary graph database migrations to the Neo4j database:

```
$ /opt/eclecticiq/platform/api/bin/manage do_graph_migrations
```

Load the dashboard

- Load the platform dashboard by running the following command(s):

```
# Run elasticdump to load the platform dashboard
$ elasticdump --input=/opt/eclecticiq/etc/kibana-dashboard.json --
output=http://localhost:9200/-kibana
```

Reload the Supervisor configuration

- Reload the Supervisor configuration by running the following command(s):

```
$ sudo supervisorctl reload
```

- Check Supervisor task statuses by running the following command(s):

```
$ supervisorctl status
```

The response should be similar to this example:

```
graph-ingestion                  RUNNING   pid 3443,  uptime 1:10:56
intel-ingestion:0                RUNNING   pid 3323,  uptime 1:11:19
intel-ingestion:1                RUNNING   pid 3336,  uptime 1:11:14
intel-ingestion:2                RUNNING   pid 3363,  uptime 1:11:09
intel-ingestion:3                RUNNING   pid 3372,  uptime 1:11:04
kibana                           RUNNING   pid 13837, uptime 9 days, 0:38:17
neo4j-batching                   RUNNING   pid 3590,  uptime 1:10:45
opentaxii                        RUNNING   pid 3662,  uptime 1:10:39
platform-api                     RUNNING   pid 2999,  uptime 1:12:47
search-ingestion                 RUNNING   pid 3513,  uptime 1:10:49
task:beat                        RUNNING   pid 3102,  uptime 1:12:39
task:enrichers                   RUNNING   pid 3110,  uptime 1:12:26
task:integrations                RUNNING   pid 3135,  uptime 1:12:13
task:providers                   RUNNING   pid 3204,  uptime 1:12:01
task:reindexing                  RUNNING   pid 3223,  uptime 1:11:48
task:utilities                   RUNNING   pid 3242,  uptime 1:11:35
```

Access the platform

To access the platform, do the following:

- Launch a web browser (recommended: Google Chrome).
- Go to the configured platform address, for example: `https://platform.host`
- On the login page, enter the appropriate credentials.

Warning: The browser may display an untrusted connection warning: add it as an exception, and then proceed to the platform.

RPM installation and configuration guide

Summary: This document guides you through the installation, setup and configuration of EclecticIQ Platform when you choose to install the product from an RPM package on a target system running CentOS or Red Hat Enterprise Linux.

Scope

This document guides you through the steps you need to carry out to complete the following tasks:

- Check and install third-party products and third-party dependencies.
- Install the platform.
- Look for and identify platform configuration and log files.
- Configure the platform and the third-party components it uses.
- Bootstrap the platform before starting it for the first time.
- Launch the platform.
- Upgrade the platform to a newer release.
- Back up your data.

Goal

After completing these tasks, you'll have achieved the following goals:

- The EclecticIQ Platform is installed, set up, and configured in the target system.
- The necessary dependencies and third-party products are installed and configured to work with the platform.
- The platform is ready for use.

Audience

This document targets the following audience:

- DevOps

- System administrators

Feedback

No one reads manuals, ever. We know.
Yet, we strive to give you clear, concise, and complete documentation that helps you get stuff done neatly.

We are committed to crafting good documentation, because life is too short for bad doc.
We appreciate your comments, and we'd love to hear from you: if you have questions or suggestions, drop us a line and share your thoughts with us!

The Product Team

Before you start

Summary: Review these system requirements before proceeding to install the platform from an RPM package.

At times throughout this document, you may need to enter commands in the terminal, in the console, or in the command line. The commands we ask you to enter are prefixed by the `$` sign, and they look like this:

```
$ run this command
```

Code and configuration examples for reference look like this:

```
{
  "this" : [
  "some awesome code",
  "or nifty configuration parameters"
  ]
}
```

About EclecticIQ Platform

EclecticIQ Platform is powered by **STIX** `(https://stixproject.github.io/)` and **TAXII** `(http://taxiiproject.github.io/about/)` open standards. It enables consolidating, analyzing, integrating, and collaborating on intel from multiple sources.

EclecticIQ Platform	key features
Feed management	Manage multiple cyber threat intelligence feeds from any source.
Enrichment	Enrich intel with external data sources, and refine it with de-duplication and pattern recognition.
Sharing	Identify threats together with partners as part of an information ecosystem.
Collaboration	Analyze and create intelligence in collaboration with other departments.
Insights	Generate insight with a high-fidelity normalized view into your intelligence.
Integration	Understand how cyber intelligence relates to your internal environment.

Hardware requirements

Hardware requirements for EclecticIQ Platform can vary depending on the target environment you plan to install the platform to. Therefore, the requirements outlined in this section are general guidelines that work in most cases, but they are not tailored to any specific situation.

Single box

Hardware requirement guidelines for EclecticIQ Platform and related dependencies installation on one target machine.

HW area	Minimum	Recommended	Notes
Environment	-	Physical machine/*rpm* install	
	-	VM/virtual appliance	
CPUs	4	8	Core count includes HT
CPU speed	2.5 GHz	2.5 GHz or faster	
Memory	16 GB	at least 32 GB	16 GB is unsuitable for production. A production environment should feature at least 32 GB memory. Consider expanding it to 64 GB when dealing with, for example, large data corpora ingestion or data-intensive graph visualizations. Monitor system memory usage to determine if your system may need more memory to operate smoothly.
Storage	SATA, 100 IOPS	SSD, 200 IOPS	Local attached storage is preferable to SAN or NAS; platform operations are write-intensive. Recommended IOPS range: 200-500
Drives	5	10	10 drives to set up 5 sets of mirrored drives (RAID 1)
Drive sizes (GB)	10, 10, 25, 50, 200	20, 20, 50, 75, 300	Each platform database should be allocated to a dedicated drive for data storage

HW area	Minimum	Recommended	Notes
Drive allocation (GB)	10	20	Root (EclecticIQ Platform + Redis)
	10	20	Log data storage
	25	50	Neo4j, graph database
	50	75	Elasticsearch, searching and indexing
	200	300	PostgreSQL, main data storage
Network	2 network interfaces	2 network interfaces	1 interface for production, the other for system management
Install size	~240 GB	~240 GB	Full install, based on VM image size

Scaling out

The easiest approach to scaling out is allocating dedicated machines to the databases. In this scenario, you install each of the following components on a separate machine:

- EclecticIQ Platform
- PostgreSQL
- Redis
- Elasticsearch
- Neo4j

To optimize read-write operations and to ensure that the storage drives are fast, set up dedicated drives per partition.

Software requirements

To correctly configure the system after installing the required products, ensure you have the following information available:

- DNS name of the host you are going to use to access the platform. Example: `platform.host`
- SSL certificate and key for the web server.
- EclecticIQ Platform login credentials.

EclecticIQ Platform default login credentials	
user	`admin`
password	`EclecticIQ2015#`

Operating systems

Supported operating systems:

- **CentOS Linux 7 (1511)** (`https://lists.centos.org/pipermail/centos-announce/2015-december/021518.html`)
- **Red Hat Enterprise Linux 7** (`https://www.redhat.com/`)

> **SELinux** (`http://selinuxproject.org/`) is supported as per release 0.13.

Repositories

- Add the following repositories to the system as root:

```
$ sudo wget https://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-9.noarch.rpm && rpm -ivh epel-release-7-9.noarch.rpm

$ sudo wget https://centos7.iuscommunity.org/ius-release.rpm && rpm -ivh ius-release.rpm

$ sudo wget https://download.postgresql.org/pub/repos/yum/9.5/redhat/rhel-7-x86_64/pgdg-centos95-9.5-3.noarch.rpm && rpm -ivh pgdg-centos95-9.5-3.noarch.rpm
```

Third-party products

Install the following software on the target system:

Third-party product	Version	Reference
Oracle Java JDK	1.8.0_71	**Oracle Java download page** (http://www.oracle.com/technetwork/java/javase/downloads/index.htm
Nginx	1.8	**Nginx web site** (http://nginx.org/en/download.html)
PostgreSQL	9.5.3	**PostgreSQL web site** (https://yum.postgresql.org/repopackages.php)
Redis	2.8.19-2.el7	**Redis web site** (http://redis.io/download)
Neo4j	2.3.1 Community	**Neo4j web site** (http://neo4j.com/download/)
unzip	-	**Install with `yum install` on CentOS/RHEL** (https://linuxmoz.com/centos-install-unzip/)
Node.js	6.x	**Node.js for CentOS and RHEL** (https://nodejs.org/en/download/package manager/#enterprise-linux-and-fedora)
Elasticsearch	2.3.3	**Elastic web site** (https://www.elastic.co/guide/en/elasticsearch/reference/2.3/setup repositories.html#_yum_dnf)
delete-by-query		**Elasticsearch plugin details** (https://www.elastic.co/guide/en/elasticsearch/plugins/2.3/plugins delete-by-query.html)
elasticdump	2.4.2	**Install globally as a Node js module** (https://www.npmjs.com/package/elasticdump)
Logstash	2.0 or higher	**Logstash install instructions** (https://www.elastic.co/guide/en/logstash/current/installing-logstash.html#_yum)
Kibana	4.5.1	**Kibana 4.5.1 download page** (https://www.elastic.co/downloads/past-releases/kibana-4-5-1)

- As per *elasticdump* version 2.4.0, we recommended installing *elasticdump* as a **global Node js module** (https://www.npmjs.com/package/elasticdump#installing).
- When carrying out maintenance and system administration activities on any of the required third-party platform components, first graciously shut down the platform, and then proceed with the other tasks.

Install via an RPM package

Summary: After checking the necessary requirements and dependencies the platform needs to work, you can proceed to install the platform.

Install from a downloaded archive

- You will receive a download link. Follow it, download the *eclecticiq-platform-x.x.x.zip .zip* archive, and save it to the target location on the host system. For example save it to the root home directory.
- Make sure you are logged in as the root user. Alternatively, run the following commands using `sudo`:

To do this…	…run this
Unzip the archive	`$ unzip eclecticiq-platform-x.x.x.zip`
Install the platform	`$ sudo yum install -y eclecticiq-platform`

- The *x.x.x* part in the archive or in the RPM package file name is replaced in the actual files with the applicable platform release number, for example *eclecticiq-platform-1.14.0.zip* or *eclecticiq-platform-1.14.0.x86_64.rpm*. Make sure you replace *x.x.x* with the appropriate release number in the file name when you execute actions on these files.
- The install command fetches and installs the required RPM platform packages.
- During the installation process, extra dependencies are automatically downloaded and installed together with the EclecticIQ RPM resources.

Install from the YUM repository

Create the YUM repository configuration

If you are upgrading or performing a fresh install of EclecticIQ Platform using the **YUM** `(http://yum.baseurl.org/)` package manager, you need to define your YUM repository configuration for the platform:

- Start the host system and log into it with the appropriate credentials.
- Create a new file, and name it `/etc/yum.repos.d/eclecticiq-platform.repo`.
- Populate the file with the following content:

```
[eclecticiq-platform]
name=eclecticiq-platform
baseurl=https://downloads.eclecticiq.com/platform
gpgcheck=1
repo_gpgcheck=0
enabled=1
username=<UPDATE_USERNAME>
password=<UPDATE_PASSWORD>
gpgkey=https://downloads.eclecticiq.com/platform/repodata/repomd.xml.key
```

Warning: Make sure you update and define user name and password values as needed.

- The installation or upgrade procedure should normally take care of removing */etc/yum.repos.d/private-eclecticiq.repo*.
  In case it does not happen automatically, you can safely remove */etc/yum.repos.d/private-eclecticiq.repo* manually:"

```
$ sudo rm -fR /etc/yum.repos.d/private-eclecticiq.repo
```

- Run the following command to upgrade the YUM repository list, so that it includes only the `eclecticiq-platform` YUM repo:

```
$ sudo yum upgrade --disablerepo=* --enablerepo=eclecticiq-platform eclecticiq-platform\*
```

If the current directory is */etc/yum.repos.d* and the `yum upgrade` command returns an error, browse to a directory you have read and write access to, and then run the command from there.
Example:

```
$ cd tmp
$ sudo yum upgrade --disablerepo=* --enablerepo=eclecticiq-platform eclecticiq-platform\*
```

Run the YUM install

To perform a **fresh install** from the YUM repository, do the following:

- Start the host system and log into it with the appropriate credentials.
- Run the following command(s):

```
$ sudo yum install -y eclecticiq-platform
```

This command fetches and installs the required RPM platform packages.
During the installation process, extra dependencies are automatically downloaded and installed together with the EclecticIQ RPM resources.

Version check

Before starting the installation, a script carries out a version check to detect any previously installed platform versions.

It is possible to upgrade the platform only sequentially, it is not possible to skip versions.
Therefore, if you want to upgrade from release 1.10.0 to 1.14.0, you need to upgrade step by step by installing all intermediate releases until 1.14.0.
Example:

```
# Upgrade from 1.10.0 to 1.10.1
$ sudo yum install -y eclecticiq-platform-1.10.1

# Upgrade from 1.10.1 to 1.11.0
$ sudo yum install -y eclecticiq-platform-1.11.0

# Upgrade from 1.11.0 to 1.12.0
$ sudo yum install -y eclecticiq-platform-1.12.0

# Upgrade from 1.12.0 to 1.13.0
$ sudo yum install -y eclecticiq-platform-1.13.0

# Upgrade from 1.13.0 to 1.14.0
$ sudo yum install -y eclecticiq-platform-1.14.0
```

If the currently installed version details are the same as the corresponding information in the upgrade version, or if the former does not immediately precedes the latter, the upgrade procedure is aborted, and an error message is displayed:

```
You currently have version ${INSTALLED_VERSION}, but you need to have ${PREV_VERSION}
installed in order to upgrade ${PACKAGE_NAME}.

exit 99
```

`${INSTALLED_VERSION}`	The currently installed version on your system. The upgrade fails because this version is too old.

`${PREV_VERSION}`	Before starting the upgrade procedure again, install this platform version on your system. The upgrade procedure works only when the old and the new versions are contiguous in sequence.
`exit 99`	The exit code accompanying the error message.

After a successful platform installation, the following message is displayed to inform you that you need to carry out a few post-installation tasks to complete the upgrade procedure:

```
You are almost there, just a few more steps before you complete the EclecticIQ
Platform upgrade procedure.
Refer to the upgrade section in the EclecticIQ Platform RPM installation and
configuration guide to learn what you should do next.
```

Just follow the next sections in this document to proceed to the configuration and bootstrapping steps.

Check SELinux

- If you are using or plan to use SELinux in the environment where the platform is installed, you should carry out this post-install check.
- If you are not using SELinux and are not planning to implement it in the environment where the platform is installed, you do not need to do anything and you can safely disregard this section.

The platform after-install script included in the RPM package attempts to automatically set the appropriate SELinux security labels for the files that are deployed to */opt/eclecticiq*.
If SELinux is not installed or if it not enabled, this process does not attempt to configure any file security labels.

If SELinux is not installed, the after-install script included in the RPM install package does not attempt to configure any SELinux file security labels for the files that are deployed to */opt/eclecticiq*.

If SELinux is installed, and the platform after-install script does not set the SELinux security labels to the applicable platform files, run the following command(s):

```
$ sudo semanage fcontext -a -t var_log_t -f d "/opt/eclecticiq/logs"
```

If SELinux policy-related errors occur, the command returns a response that can be similar to this example:

```
SELinux:  Could not downgrade policy file /etc/selinux/targeted/policy/policy.29,
searching for an older version.
SELinux:  Could not open policy file <= /etc/selinux/targeted/policy/policy.29:  No
such file or directory
/sbin/load_policy:  Can't load policy:  No such file or directory
libsemanage.semanage_reload_policy: load_policy returned error code 2.
```

The response provides more context about the affected files and the reasons why it was not possible to set the security labels.

If SELinux is installed, check if it is enabled or disabled. Run the following command(s):

```
$ sudo sestatus -v
```

If SELinux is disabled, the response includes the following line:

```
SELinux status: disabled
```

You can check also which SELinux mode is currently active. Run the following command(s):

```
$ sudo getenforce
```

The allowed modes are `enforcing`, `permissive`, and `disabled`.
The active mode may not be the same as the `SELINUX` value defined in the SELinux global configuration file:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are
protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

This can happen after changing and saving **SELinux global configuration file** `(https://selinuxproject.org/page/configurationfiles)`, and before executing a system reboot for the changes to become effective.

SELinux is not installed

If SELinux is not installed on the target system, do the following:

- After completing the platform installation, install and enable SELinux.
- To set the correct security contexts, execute the following script:

```
BASE_PATH="/opt/eclecticiq"

if [ -x "$(command -v semanage)" ]; then

    SELINUX_MODE=$(getenforce)

    if ! [ $SELINUX_MODE == "Disabled" ]; then

        semanage fcontext -a -t etc_t "$BASE_PATH/etc(/.*)?"
        semanage fcontext -a -t etc_t "$BASE_PATH/etc-extras(/.*)?"

        semanage fcontext -a -t httpd_config_t "$BASE_PATH/etc/nginx(/.*)?"
        semanage fcontext -a -t httpd_config_t "$BASE_PATH/etc-extras/nginx(/.*)?"

        # By default, newly created files and directories inherit the SELinux type
        # of the corresponding parents, so that log files have the correct type.
        # However, we do not want to relabel existing logs.
        semanage fcontext -a -t var_log_t -f d "$BASE_PATH/logs"

        restorecon -RF $BASE_PATH

        echo "SELinux security labels configured."
    else
        echo "SELinux is not enabled. Security labels won't be configured."
    fi
else
    echo "SELinux is not installed. Security labels won't be configured."
fi
```

- You may need to reboot the system for the changes to become effective.

SELinux is installed but it is not enabled

If SELinux is installed on the target system but it is not enabled, do the following:

- Enable SELinux, either by editing its configuration file, and then by rebooting the system, or by running one of the following commands:

```
# Set SELinux to permissive mode
$ sudo setenforce 0

# Set SELinux to enforcing mode
$ sudo setenforce 1
```

- Create the following bash script:

```
BASE_PATH="/opt/eclecticiq"

if [ -x "$(command -v semanage)" ]; then

    SELINUX_MODE=$(getenforce)

    if ! [ $SELINUX_MODE == "Disabled" ]; then

        semanage fcontext -a -t etc_t "$BASE_PATH/etc(/.*)?"
        semanage fcontext -a -t etc_t "$BASE_PATH/etc-extras(/.*)?"

        semanage fcontext -a -t httpd_config_t "$BASE_PATH/etc/nginx(/.*)?"
        semanage fcontext -a -t httpd_config_t "$BASE_PATH/etc-extras/nginx(/.*)?"

        # By default, newly created files and directories inherit the SELinux type
        # of the corresponding parents, so that log files have the correct type.
        # However, we do not want to relabel existing logs.
        semanage fcontext -a -t var_log_t -f d "$BASE_PATH/logs"

        restorecon -RF $BASE_PATH

        echo "SELinux security labels configured."
    else
        echo "SELinux is not enabled. Security labels won't be configured."
    fi
else
    echo "SELinux is not installed. Security labels won't be configured."
fi
```

- Save it, make it executable, and then run it.
- You may need to reboot the system for the changes to become effective.

Config and log files

Summary: An overview of all platform configuration, log, and manifest files for system administrators.

Configuration, log and manifest files

The EclecticIQ Platform uses several configuration files to store platform settings you can edit and fine-tune to adapt the behavior of the platform to your system.
Log files record platform events; they hold a history of the platform activities that can provide meaningful context, for example when investigating the possible root causes of a problem.
Manifest files contain metadata that help identify the product like the source/origin of the package containing the platform and its components, release reference number, and version information.

This section describes where the platform configuration, log, and manifest files are stored, and what kind of information each file holds.

Configuration files

To get a list with the platform configuration files, run the following command(s):

```
# Returns only core config files
$ find /opt/eclecticiq/etc/eclecticiq/ -type f

# Returns platform and third-party components config files
$ find /opt/eclecticiq/etc/ -type f

# Returns third-party config files to use for reference as boilerplates
$ find /opt/eclecticiq/etc-extras/ -type f
```

The response returns a list with the following files:

```
# Core platform settings
/opt/eclecticiq/etc/eclecticiq/opentaxii.yml
/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/etc/eclecticiq/proxy_url

# Dashboard settings
/opt/eclecticiq/etc/kibana-dashboard.json

# Logstash log aggregation settings
```

```
/opt/eclecticiq/etc/logstash/conf.d/filters.conf
/opt/eclecticiq/etc/logstash/conf.d/input.conf
/opt/eclecticiq/etc/logstash/conf.d/neo4j-batching.conf
/opt/eclecticiq/etc/logstash/conf.d/opentaxii.conf
/opt/eclecticiq/etc/logstash/conf.d/output.conf
/opt/eclecticiq/etc/logstash/conf.d/platform-api.conf
/opt/eclecticiq/etc/logstash/conf.d/platform-ui.conf

# Logstash heap size
/opt/eclecticiq/etc/sysconfig/logstash

# Web server, proxy and port settings
/opt/eclecticiq/etc/nginx/conf.d/platform.conf

# Platform components ini file settings: ingestion, search, graph, taxii, api, tasks
/opt/eclecticiq/etc/supervisord.d/graph-ingestion.ini
/opt/eclecticiq/etc/supervisord.d/intel-ingestion.ini
/opt/eclecticiq/etc/supervisord.d/neo4j-batching.ini
/opt/eclecticiq/etc/supervisord.d/opentaxii.ini
/opt/eclecticiq/etc/supervisord.d/platform-api.ini
/opt/eclecticiq/etc/supervisord.d/search-ingestion.ini
/opt/eclecticiq/etc/supervisord.d/task-workers.ini

# Kibana config file
/opt/eclecticiq/etc-extras/kibana.yml

# Redis message broker config file
/opt/eclecticiq/etc-extras/redis.conf

# Elasticsearch config files
/opt/eclecticiq/etc-extras/elasticsearch/elasticsearch.yml
/opt/eclecticiq/etc-extras/elasticsearch/logging.yml

# Neo4j config files
/opt/eclecticiq/etc-extras/neo4j/neo4j.properties
/opt/eclecticiq/etc-extras/neo4j/neo4j-server.properties
/opt/eclecticiq/etc-extras/neo4j/neo4j-wrapper.conf

# Nginx web server config file
/opt/eclecticiq/etc-extras/nginx/nginx.conf

# Postfix email server config file
/opt/eclecticiq/etc-extras/postfix/main.cf

# Kibana and Neo4j ini files
/opt/eclecticiq/etc-extras/supervisord/kibana.ini
/opt/eclecticiq/etc-extras/supervisord/neo4j-console.ini
```

The following table gives an overview of what information each configuration file holds.

File name and location	D

File name and location	D
`/opt/eclecticiq/etc/eclecticiq/platform_settings.py`	Contains core platform settings like security k URLs ponting to external components Celery-
`/opt/eclecticiq/etc/eclecticiq/opentaxii.yml`	Contains **OpenTAXII** (`https://opentaxii.` parameters like URL and port for the service, message broker to use.
`/opt/eclecticiq/etc/eclecticiq/proxy_url`	Contains the IP addresses and host names th comma-separated. The no-proxy list needs to `127.0.0.1,localhost`.
`/opt/eclecticiq/etc/kibana-dashboard.json`	Contains the configuration defining the Kibana based GUI.
`/opt/eclecticiq/etc/logstash/conf.d/platform-api.conf`	Defines the locations of the log files storing lo API, intel ingestion, and tasks.
`/opt/eclecticiq/etc/logstash/conf.d/platform-ui.conf`	Defines the locations of the log files storing lo is the web server; if it is not working normally unavailable.
`/opt/eclecticiq/etc/logstash/conf.d/input.conf`	Elasticsearch Curator input configuration file. (`https://www.elastic.co/guide/en/elas` manages Elasticsearch indices.
`/opt/eclecticiq/etc/logstash/conf.d/output.conf`	Defines the Elasticsearch cluster and the data to Elasticsearch.
`/opt/eclecticiq/etc/logstash/conf.d/filters.conf`	Defines filters as needed to select specific ind
`/opt/eclecticiq/etc/logstash/conf.d/neo4j-batching.conf`	Configuration file of the `neo4j-batching` proc
`/opt/eclecticiq/etc/logstash/conf.d/opentaxii.conf`	Defines the locations of the log files storing lo is the web server OpenTAXII relies on.
`/opt/eclecticiq/etc/nginx/conf.d/platform.conf`	Defines the platform configuration the Nginx w key, ports, endpoints to make available servic documentation, and so on.
`/opt/eclecticiq/etc/sysconfig/logstash`	INI configuration file defining the Logstash hea GB)
`/opt/eclecticiq/etc/supervisord.d/platform-api.ini`	Supervisor INI configuration file to start the pla
`/opt/eclecticiq/etc/supervisord.d/graph-ingestion.ini`	Supervisor INI configuration file to start the gra
`/opt/eclecticiq/etc/supervisord.d/intel-ingestion.ini`	Supervisor INI configuration file to start the int

File name and location	D
`/opt/eclecticiq/etc/supervisord.d/search-ingestion.ini`	Supervisor INI configuration file to start the El
`/opt/eclecticiq/etc/supervisord.d/neo4j-batching.ini`	Initializes the `neo4j-batching` process. By d
`/opt/eclecticiq/etc/supervisord.d/opentaxii.ini`	Supervisor INI configuration file to start Open incoming and outgoing feeds.
`/opt/eclecticiq/etc/supervisord.d/task-workers.ini`	Supervisor INI configuration file to start the Ce `integrations`, `enrichers`, `utilities`, and
`/opt/eclecticiq/etc-extras/kibana.yml`	Kibana configuration file. Check it and edit or newer version.
`/opt/eclecticiq/etc-extras/redis.conf`	Defines the configuration for the Redis messa snapshot autosave time intervals, and so on.
`/opt/eclecticiq/etc-extras/elasticsearch/elasticsearch.yml`	Elasticsearch configuration file. Check it and e Elasticsearch to a newer version.
`/opt/eclecticiq/etc-extras/elasticsearch/logging.yml`	Configures logging and sets the logging level
`/opt/eclecticiq/etc-extras/neo4j/neo4j.properties`	Defines the maximum size for page cache an
`/opt/eclecticiq/etc-extras/neo4j/neo4j-server.properties`	Defines the configuration options of the Neo4j communication port, and so on.
`/opt/eclecticiq/etc-extras/neo4j/neo4j-wrapper.conf`	Defines the Java settings related to Neo4j like database. Default min. and max. values: 4096
`/opt/eclecticiq/etc-extras/nginx/nginx.conf`	Defines the configuration options for the Ngin
`/opt/eclecticiq/etc-extras/supervisord/kibana.ini`	Initializes the `kibana` process. By default, the
`/opt/eclecticiq/etc-extras/supervisord/neo4j-console.ini`	Initializes the `neo4j-console` process, that is start automatically.
`/etc/supervisord.conf`	Contains the **Supervisor configuration** `(htt` `supervisord` and `supervisorctl`. This file sl `(http://supervisord.org/configuration` enabling the `supervisord` server to reply to s
`/var/lib/pgsql/9.5/data/postgresql.conf`	**PostgreSQL configuration file** `(https://w` `setting.html)`.
`/opt/eclecticiq/etc-extras/postfix/main.cf`	**Postfix configuration file** `(http://www.pos` configuring email addreses through the GUI, y and SMTP in this file.

Log files

To get a list with the log files created by the platform and its components, run the following command(s):

```
$ find /opt/eclecticiq/logs -type f

$ find /var/log -type f
```

The response returns a list with the following files:

```
# Platform core services and components logs:
# API, ingestion, graph, OpenTAXII, Celery-managed task workers
/opt/eclecticiq/logs/graph-ingestion.log
/opt/eclecticiq/logs/intel-ingestion.log
/opt/eclecticiq/logs/kibana.log
/opt/eclecticiq/logs/neo4j-batching.log
/opt/eclecticiq/logs/neo4j.log
/opt/eclecticiq/logs/opentaxii.log
/opt/eclecticiq/logs/platform-api.log
/opt/eclecticiq/logs/search-ingestion.log
/opt/eclecticiq/logs/task-beat.log
/opt/eclecticiq/logs/task-worker-enrichers.log
/opt/eclecticiq/logs/task-worker-integrations.log
/opt/eclecticiq/logs/task-worker-providers.log
/opt/eclecticiq/logs/task-worker-reindexing.log
/opt/eclecticiq/logs/task-worker-utilities.log

# Logstash log data aggregation logs
/var/log/logstash/logstash.err
/var/log/logstash/logstash.log
/var/log/logstash/logstash.stdout

# Elasticsearch search indexing logs
/var/log/elasticsearch/intel.log
/var/log/elasticsearch/intel_deprecation.log
/var/log/elasticsearch/intel.log.YYY-MM-DD.log
/var/log/elasticsearch/intel_index_indexing_slowlog.log
/var/log/elasticsearch/intel_index_search_slowlog.log

# Nginx web server logs
/var/log/nginx/access.log
/var/log/nginx/eclecticiq-platform-ui-nginx-access.log
/var/log/nginx/eclecticiq-platform-ui-nginx-errors.log
/var/log/nginx/error.log

# PostgreSQL intel database log
/var/log/postgresql/postgresql-YYYY-MM-DD.log

# Redis message broker log
/var/log/redis/redis.log
```

The following table gives an overview of what information each log file holds.

File name and location	Description
`/opt/eclecticiq/logs/platform-api.log`	Platform log file. It logs core platform information.
`/opt/eclecticiq/logs/graph-ingestion.log`	Neo4j graph database log file. It logs information on graph database migrations, indices, and graph ingestion queue.

File name and location	Description
/opt/eclecticiq/logs/intel-ingestion.log	Intel ingestion log file. It logs information on ingestion events, as well as ingested batches and blobs.
/opt/eclecticiq/logs/search-ingestion.log	Search indexing log file. It logs information on Elasticsearch data indexing.
/opt/eclecticiq/logs/kibana.log	It logs information on Kibana dashboard like connection and availability.
/opt/eclecticiq/logs/neo4j.log	Neo4j graph database log file. It logs information on Neo4j server and database operation.
/opt/eclecticiq/logs/opentaxii.log	It logs OpenTAXII server log information.
/opt/eclecticiq/logs/task-beat.log	It logs heartbeat timestamp information. The heartbeat interval is 30 seconds.
/opt/eclecticiq/logs/task-worker-reindexing.log	It lists synced enricher tasks, intel providers, feed transport types, and platform utility tasks. Indexing and syncing go through Redis and Celery.
/opt/eclecticiq/logs/task-worker-enrichers.log	It lists enricher tasks, intel providers, feed transport types, and platform utility tasks running in the background.
/opt/eclecticiq/logs/task-worker-integrations.log	It lists integrations like enricher tasks, intel providers, and platform utility tasks running in the background.
/opt/eclecticiq/logs/task-worker-providers.log	It lists integrations like enricher tasks, intel providers, and platform utility tasks running in the background.
/opt/eclecticiq/logs/task-worker-utilities.log	It lists integrations like enricher tasks, intel providers, and platform utility tasks running in the background.
/var/log/logstash/logstash.err	It logs Logstash errors and error messages.

File name and location	Description
`/var/log/logstash/logstash.log`	It logs Logstash events.
`/var/log/logstash/logstash.stdout`	Standard output format for Logstash log information.
`/var/log/elasticsearch/intel.log`	Elasticsearch log file. It logs Elasticsearch events like initialization, startup, designated Elasticsearch cluster, and so on.
`/var/log/elasticsearch/intel.log.YYYY-MM-DD.log`	Elasticsearch log file for a specific date. *YYYY-MM-DD* (year, month, day) in the file name is replaced by the date the log information refers to. It logs Elasticsearch events like initialization, startup, designated Elasticsearch cluster, and so on.
`/var/log/elasticsearch/intel_index_indexing_slowlog.log`	Elasticsearch log file. It logs Elasticsearch indexing information.
`/var/log/elasticsearch/intel_index_search_slowlog.log`	Elasticsearch log file. It logs Elasticsearch search index information.
`/var/log/postgresql/postgresql-YYYY-MM-DD.log`	PostgreSQL log file. It logs PostgreSQL database intel ingestion information.
`/var/log/redis/redis.log`	Redis log file. It logs message broker event information about memory usage during copy-write operations and data saving to the database.
`/var/log/nginx/access.log`	Nginx log file. It logs web server access information.
`/var/log/nginx/eclecticiq-platform-ui-nginx-access.log`	Nginx log file. It logs web server access information related to the platform web-based GUI.
`/var/log/nginx/eclecticiq-platform-ui-nginx-errors.log`	Nginx log file. It logs web server error information related to the platform web-based GUI.
`/var/log/nginx/error.log`	Nginx log file. It logs web server error information.

Manifest files

To get a list with the manifest files created by the platform and its components during the installation operation, run the following command(s):

```
$ find /opt/eclecticiq/manifests -type f
```

The response returns a list with the following files:

```
# Platform manifest files:
/opt/eclecticiq/manifests/platform-api.mf
/opt/eclecticiq/manifests/platform-database.mf
/opt/eclecticiq/manifests/platform-ui.mf
/opt/eclecticiq/manifests/opentaxii.mf
/opt/eclecticiq/manifests/opentaxii-platform-apis.mf
```

The following table gives an overview of the metadata each manifest file holds.

File name and location	Description
`/opt/eclecticiq/manifests/platform-api.mf`	Manifest file with platform API metadata like package source/origin, release reference number, and version information.
`/opt/eclecticiq/manifests/platform-database.mf`	Manifest file with platform database release metadata like package source/origin.
`/opt/eclecticiq/manifests/platform-ui.mf`	Manifest file with platform GUI metadata like package source/origin, release reference number, and version information.
`/opt/eclecticiq/manifests/opentaxii.mf`	Manifest file with OpenTAXII metadata like package source/origin and version information.
`/opt/eclecticiq/manifests/opentaxii-platform-apis.mf`	Manifest file with OpenTAXII metadata like package source/origin, release reference number, and version information.

Default users

Summary: An overview of the default user profiles that are created during a clean platform installation.

During the installation procedure, several default users profiles are created at platform level, as well as host operating system level, to access and manage third-party components and processes.

These users receive a standard set of user rights and permissions to allow them to carry out their tasks. They interact only with the component(s) they manage and control. In other words, these users and groups are organized in separate compartments, where each user is responsible for one or more specific, and closely related, tasks.

User	Group	Sudo	Component	Description	Home dir
`eclecticiq`	`eclecticiq`	✖	Celery workers and task runners, graph ingestion, intel ingestion, search ingestion	Platform user responsible for operational tasks like accessing Celery tasks, writing data to the graph ingestion storage location, and accessing the TAXII service	`/opt/eclecticiq`
`elasticsearch`	`elasticsearch`	✖	Elasticsearch search and indexing database	Search and indexing database user	`/home/elasticsearch`
`logstash`	`logstash`	✖	Logstash log aggregator	Log aggregator user	`/opt/logstash`
`neo4j`	`neo4j`	✖	Neo4j graph database	Graph database user	`/usr/share/neo4j`
`nginx`	`nginx`	✖	Nginx web server	Web server user	`/var/cache/nginx`

User	Group	Sudo	Component	Description	Home dir
`postgres`	`postgres`	✖	PostgreSQL database	Database user, can access the default `platform-api` database	`/var/lib/pgsql`
`redis`	`redis`	✖	Redis server, message broker and queue manager	Redis database and message broker user	`/var/lib/redis`

To view platform user profiles, go to **System > User management**, and then select **Users**, **Groups**, **Roles** and **Permissions** (non-editable) to display the corresponding overview.

Configure the platform

Summary: Configure the third-party products the platform interacts with, and then update the platform settings to reflect the configuration changes.

After installing the platform, you can proceed to configuring it, along with the required third-party components. Let's start by creating the main data storage, setting up the web server, and configuring graph, indexing, and search:

- Create the database (PostgreSQL)
- Configure the web server (Nginx)
- Configure the data structure store (Redis)
- Configure the search server (Elasticsearch)
- Configure the log aggregator (Logstash)
- Configure the graph database (Neo4j)
- Update the platform settings.

Create the database

- Connect to PostgreSQL with root privileges, and access it with the `postgres` user name:

```
$ su - postgres
$ psql
```

- In PostgreSQL, do the following:
  - **Create** `(http://www.postgresql.org/docs/current/static/tutorial-createdb.html)` a new database (in the example: *platform*).
  - Create a new user, and assign them a password (in the example: *test*/*test*, respectively).
  - Grant the new user full privileges on the newly created database.

```
CREATE DATABASE platform;
CREATE USER test WITH PASSWORD 'test';
GRANT ALL PRIVILEGES ON DATABASE platform to test;
```

Add the database to the platform settings

- In the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* platform configuration file, update the `SQLALCHEMY_DATABASE_URI` parameter with the designated database URI for your environment:

```
SQLALCHEMY_DATABASE_URI = "postgresql://<username>:<password>@<host>:<port>/<database-name>"
```

`username`	Replace this placeholder with the user name of the user that can access the database. Example: `test`.
`password`	Replace this placeholder with the corresponding user password. Example: `test`.
`host`	Replace this placeholder with the host name identifying the location where the database is hosted. Example: `127.0.0.1`.
`port`	The database access port. Default value: `5432`.
`database-name`	The assigned name to identify the database. Example: `platform`.

Set the database authentication method

- Open the *pg_hba.conf* configuration file in a text editor:

```
$ sudo nano /var/lib/pgsql/9.5/data/pg_hba.conf
```

- Set the authentication method to `trust` for the `local` type, and to `md5` for the `host` type to request an MD5-encrypted password:

```
# TYPE    DATABASE    USER       ADDRESS     METHOD
  local   all         test                   trust
  host    all         all        samenet     md5
```

Restart the database service

- Restart the PostgreSQL service:

```
$ sudo systemctl restart postgresql-9.5.service
```

or:

```
$ sudo service postgresql-9.5 restart
```

- Verify that user and database are successfully created:

```
$ psql -U test -d platform -W
```

Configure Nginx

Nginx `(http://nginx.org/en/download.html)` is the web server we are setting up for the platform.

To configure Nginx, do the following:

- Go to */etc/nginx*.
  You may need to access the resource by prefixing `sudo su`, so that you can obtain root privileges.
- Back up */etc/nginx/nginx.conf*, and replace it with the reference *nginx.conf* file shipped with the platform:

```
$ sudo su
$ cp /opt/eclecticiq/etc-extras/nginx/nginx.conf /etc/nginx/nginx.conf
```

- Edit */etc/nginx/nginx.conf* to reflect your actual configuration:

```
$ sudo su
$ sudo nano /etc/nginx/nginx.conf
```

Set Nginx user and group

Check the following parameters and their settings, since they affect interoperability between the platform and Nginx:

- The `user` should be set to `nginx nginx` for user name and group name, respectively:

```
user    nginx nginx;
```

Check access log format

Verify the Nginx **access log** `(https://www.nginx.com/resources/admin-guide/logging-and-monitoring/)` format to make sure Nginx can recognize and consume the *expected format for web server logs*.

- The `log_format` directive holds the configuration parameters for the **log format** `(https://nginx.org/en/docs/http/ngx_http_log_module.html#log_format)`.
  As per Nginx 1.8, the following example shows the default JSON format for web server logs:

```
log_format json '{'
                    '"remote_addr": "$remote_addr",'
                    '"remote_user": "$remote_user",'
                    '"time_local": "$time_local",'
                    '"request": "$request",'
                    '"status": $status,'
                    '"body_bytes_sent": $body_bytes_sent,'
                    '"http_referer": "$http_referer",'
                    '"http_user_agent": "$http_user_agent",'
                    '"http_x_forwarded_for": "$http_x_forwarded_for"'
                    '}';
```

- Make sure the **access_log** (https://nginx.org/en/docs/http/ngx_http_log_module.html#access_log) format value matches the corresponding value specified in `log_format <format_name>`.
  The default Nginx log format for EclecticIQ Platform is `json`:

```
access_log /var/log/nginx/access.log json;
```

- To enable Nginx to pick up and start the platform configuration, copy *platform.conf*
  - from */opt/eclecticiq/etc/nginx/conf.d/platform.conf*
  - to */etc/nginx/conf.d*

```
$ cp /opt/eclecticiq/etc/nginx/conf.d/platform.conf /etc/nginx/conf.d/platform.conf
```

- Open *platform.conf* in a text editor:

```
$ sudo nano /etc/nginx/conf.d/platform.conf
```

- Look for the `server_name` property.
- Replace `server_name <default_server_name>;` with the appropriate platform host name for your environment:

```
// Default server name value:
server_name <default_server_name>;

// Change it to your platform host name:
server_name <platform_server_name>;
```

- Make sure that the SSL certificate paths point to the correct locations where your certificates are stored. Example:

```
ssl_certificate      /etc/pki/tls/certs/local.iw.crt;
ssl_certificate_key  /etc/pki/tls/private/local.iw.key;
```

- Enable the Nginx service to start at system bootup:

```
$ sudo systemctl enable nginx
```

- **Reload** (http://nginx.org/en/docs/beginners_guide.html#control) the Nginx configuration and start the new worker process with a new/updated configuration:

```
$ sudo service nginx reload
```

- Start the Nginx web server:

```
$ sudo systemctl start nginx
```

or:

```
$ sudo service nginx start
```

Configure Redis

To configure Redis, do the following:

- Go to */etc*.
  You may need to access the resource by prefixing `sudo su`, so that you can obtain root privileges.
- Back up */etc/redis.conf*, and replace it with the reference *redis.conf* file shipped with the platform:

```
$ sudo su
$ cp /opt/eclecticiq/etc-extras/redis.conf /etc/redis.conf
```

For reference, look up a copy of the **self-documented file** (https://raw.githubusercontent.com/antirez/redis/2.8/redis.conf), and the **Redis configuration documentation** (https://redis.io/topics/config).

Check the Redis configuration

- Edit */etc/redis.conf* to reflect your actual configuration.

- Check the following parameters and their settings, since they affect interoperability between the platform and Redis:
  - `port 6379`: this is the default port for Redis.
  - `dbfilename dump.rdb`: the default file name of the database Redis saves snapshots to.
  - `dir /<some_dir>/<redis_snapshot>`: sets the location where Redis stores the snapshot database.

Check permissions

Make sure the target directory to save application data to exists before setting it in the configuration file, and that it is accessible to read and write data. If necessary, do the following:

- Create the target directory:

```
$ sudo mkdir /<some_dir>/<target_dir>
```

- Make sure the correct user owns the target directory to access it in read and write modes.
  For example, if you set the data storage location to `/media/redis`, the `redis` user should own it:

```
$ sudo chown -R redis:redis /media/redis
```

Check the platform settings

- Open */opt/eclecticiq/etc/eclecticiq/platform_settings.py*, and verify that `REDIS_URL` points to the correct location in your environment:

```
REDIS_URL="redis://<redis.host>:<redis.port>"
```

- To verify that Redis is correctly installed, do the following:

```
$ type redis-server

$ type redis-cli
```

- To start Redis, run the following command(s):

```
$ sudo service redis start
```

- To check if Redis is running, run the following command(s):

```
$ sudo service redis status
```

- To verify that Redis is working properly, run the following command(s):

```
# Ping Redis to ask how it is doing
$ redis-cli ping

# Redis responds to confirm everything is ok
PONG
```

Configure Elasticsearch

To configure Elasticsearch, do the following:

- Go to */etc/elasticsearch*.
  You may need to access the resource by prefixing `sudo su`, so that you can obtain root privileges.
- Back up */etc/elasticsearch/elasticsearch.yml*, and replace it with the reference *elasticsearch.yml* file shipped with the platform:

```
$ sudo su
$ cp /opt/eclecticiq/etc-extras/elasticsearch/elasticsearch.yml
/etc/elasticsearch/elasticsearch.yml
```

- Edit */etc/elasticsearch/elasticsearch.yml* to reflect your actual configuration:

```
$ sudo su
$ nano /etc/elasticsearch/elasticsearch.yml
```

Check the following parameters and their settings, since they affect interoperability between the platform and Elasticsearch:

- Set `path.data` to the location where Elasticsearch stores its data.

Check permissions

Make sure the target directory to save application data to exists before setting it in the configuration file, and that it is accessible to read and write data. If necessary, do the following:

- Create the target directory:

```
$ sudo mkdir /<some_dir>/<target_dir>
```

- Make sure the correct user owns the target directory to access it in read and write modes.
  For example, if you set the data storage location to `/media/elasticsearch`, the `elasticsearch` user should own it:

```
$ sudo chown -R elasticsearch:elasticsearch /media/elasticsearch
```

Disable dynamic scripting

- You should disable **dynamic scripting** (https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-scripting.html#enable-dynamic-scripting) by either removing the `script.inline` property from the configuration file, or by setting it to `false`:

```
script.inline: false
script.indexed: true
```

Reference *elasticsearch.yml* file:

```
config:

  cluster.name: intel

  discovery.zen.ping.multicast: false

  index.number_of_replicas: 0

  index.number_of_shards: 1

  node.local: true

  path:

    data: /<path_to>/<elasticsearch_index_datastore>

  script.indexed: true

  script.inline: false
```

Check the Elasticsearch URL

- Open */opt/eclecticiq/etc/eclecticiq/platform_settings.py*, and verify that `SEARCH_URL` points to the correct location corresponding to the Elasticsearch instance:

```
SEARCH_URL="http://<elasticsearch.host>:<elasticsearch.port>"
```

Install the required plugins

- Install the following plugins:
  - *Optional* — **mobz/elasticsearch-head** (https://github.com/mobz/elasticsearch-head)
  - *Optional* — **codelibs/elasticsearch-reindexing** (https://github.com/codelibs/elasticsearch-reindexing)
  - **Required** — **delete-by-query** (https://www.elastic.co/guide/en/elasticsearch/plugins/2.3/plugins-delete-by-query.html)
    To install *delete-by-query*, run the following command(s):

```
$ /usr/share/elasticsearch/bin/plugin install delete-by-query
```

Configure Logstash

Logstash configuration files are copied automatically to the correct location during the post-install step.
To verify that the configuration files exist in the target destination, run the following command(s):

```
$ ls -l /opt/eclecticiq/etc/logstash/conf.d/
```

These are the Logstash configuration files related to the platform:

```
filters.conf
input.conf
neo4j-batching.conf
opentaxii.conf
output.conf
platform-api.conf
platform-ui.conf
```

The following files control the **Logstash processing pipeline** (https://www.elastic.co/guide/en/logstash/current/pipeline.html):

```
input.conf
filters.conf
output.conf
```

For further details, see the how-to article on addressing logging issues in Kibana and Logstash configurations.

Configure Kibana

- To configure Kibana, simply replace */opt/kibana/config/kibana.yml* with the provided */opt/eclecticiq/etc-extras/kibana.yml*.
  The default settings in the */opt/eclecticiq/etc-extras/kibana.yml* configuration file should work in your environment without requiring any changes.

```
$ cp /opt/eclecticiq/etc-extras/kibana.yml /opt/kibana/config/kibana.yml
```

Kibana uses an index in Elasticsearch to store saved searches, visualizations, and dashboards.
The default property/value pair defining the index in the *kibana.yml* configuration file is `kibana_index: -kibana`.

The default parameters and values should not need any editing:

```
port: 5601
host: "0.0.0.0"
elasticsearch_url: "http://127.0.0.1:9200"
elasticsearch_preserve_host: true
kibana_index: "-kibana"
server.basePath: "/api/kibana"
default_app_id: "discover"
request_timeout: 300000
shard_timeout: 0
verify_ssl: false
bundled_plugin_ids:
 - plugins/dashboard/index
 - plugins/discover/index
 - plugins/doc/index
 - plugins/kibana/index
 - plugins/markdown_vis/index
 - plugins/metric_vis/index
 - plugins/settings/index
 - plugins/table_vis/index
 - plugins/vis_types/index
 - plugins/visualize/index
```

Kibana 4.5.x has an **issue** `(https://github.com/elastic/kibana/issues/6730)` that causes root owned files in */opt/kibana/optimize/*. This can prevent Kibana from running.
It should be fixed in version 4.6.0.

To work around it, you can change permissions so that the `kibana` user owns or has read/write access to the */opt/kibana/optimize/* directory.
To do so, run the following command(s):

```
$ sudo chown -Rvf kibana:kibana /opt/kibana/optimize/*
```

Configure Neo4j

To configure Neo4j, do the following:

- Go to */opt/eclecticiq/etc-extras/neo4j*.
- Copy the Neo4j configuration files shipped with the platform to */etc/neo4j*, so that they replace the default configuration files created during the Neo4j installation:

```
$ cp /opt/eclecticiq/etc-extras/neo4j/* /etc/neo4j/
```

The action should copy the following Neo4j configuration files to the destination directory:

```
neo4j-server.properties
neo4j-wrapper.conf
neo4j.properties
```

Check Neo4j connectivity

Check the main connectivity properties to verify that Neo4j can communicate with other components.

Check neo4j-server.properties

- Open the *neo4j-server.properties* configuration file:

```
$ nano /etc/neo4j/neo4j-server.properties
```

- Make sure that the following properties are set to the corresponding values:

```
org.neo4j.server.webserver.address=0.0.0.0
org.neo4j.server.webserver.port=7474
org.neo4j.server.database.location=/<some_dir>/graph.db
dbms.security.auth_enabled=False
```

`graph.db` is the default name of the graph database file.
Make sure the directory where it is stored exists before setting it in this configuration file, and that Neo4j can read and write data to it.

Inspect the amount of allocated memory: Neo4j purrs when you give it plenty of memory to work with.

- Depending on your system resources, allocate at least 4 GB to Neo4j.
- Never set it below 1 GB.

To set the minimum and maximum memory values for Neo4j, do the following:

Check neo4j-wrapper.conf

- Open the *neo4j-wrapper.conf* configuration file:

```
$ nano /etc/neo4j/neo4j-wrapper.conf
```

- Make sure that the following properties are set to the corresponding values:

```
wrapper.java.initmemory=1024
wrapper.java.maxmemory=4096
```

- The recommended value for `wrapper.java.initmemory` is at least 4096 MB or more, if possible. The minimum value should never be lower than 1024 MB.
- The recommended value for `wrapper.java.maxmemory` is at least 4096 MB or more, if possible. This value should never be lower than `wrapper.java.initmemory`.

Check neo4j.properties

Set the size of the page cache to `5G`:

- Open the *neo4j.properties* property file:

```
$ nano /etc/neo4j/neo4j.properties
```

- Make sure the `dbms.pagecache.memory` property is set to `5G`:

```
dbms.pagecache.memory=5G
```

If this parameter is not configured, Neo4j defaults to using 50% of the available system RAM for page caching.

Check permissions

Make sure the target directory to save application data to exists before setting it in the configuration file, and that it is accessible to read and write data. If necessary, do the following:

- Create the target directory:

```
$ sudo mkdir /<some_dir>/<target_dir>
```

- Make sure the correct user owns the target directory to access it in read and write modes. For example, if you set the data storage location to `/media/neo4j`, the `neo4j` user should own it:

```
$ sudo chown -R neo4j:neo4j /media/neo4j
```

Check the Neo4j URL

- Open */opt/eclecticiq/etc/eclecticiq/platform_settings.py*, and verify that `NEO4J_URL` points to the correct Neo4j instance in your environment:

```
NEO4J_URL="http://<Neo4j.host>:<Neo4j.port>"
```

Update the platform settings

At this point, you have set up the third-party components. You can now proceed to update the platform settings.

- Review the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* platform settings file to make sure it reflects your environment.
  The following example serves as a guideline:

```
PLATFORM_MANIFESTS_DIR = '/opt/eclecticiq/manifests'
COMPONENT_SHARED_SECRET = "<component_secret_key_value>"
SECRET_KEY = "<secret_key_value>"
PROXY_URL_FILE_PATH = '/opt/eclecticiq/etc/eclecticiq/proxy_url'
SQLALCHEMY_DATABASE_URI = "postgresql://<username>:<password>@localhost:5432/platform"
SQLALCHEMY_POOL_SIZE = 20
SQLALCHEMY_MAX_OVERFLOW = 20
DISCOVERY_WORKSPACE_SIZE_LIMIT = 500
DISCOVERY_SEARCH_LIMIT = 1000
DISCOVERY_RESULTS_LIMIT = 500
INGESTION_QUEUE_NAME = 'queue:ingestion:inbound'
ENRICHMENT_QUEUE_NAME = 'queue:enrichment:inbound'
GRAPH_QUEUE_NAME = 'queue:graph:inbound'
GRAPH_INGESTION_SPOOL_DIRECTORY = '/opt/eclecticiq/tmp/gi-storage'
SEARCH_QUEUE_NAME = 'queue:search:inbound'
REDIS_URL = 'redis://127.0.0.1:6379/'
KIBANA_VERSION = '4.5.1'
KIBANA_URL = 'http://127.0.0.1:5601'
NEO4J_URL = 'http://127.0.0.1:7474'
NEO4J_BATCHING_URL = 'http://127.0.0.1:4008'
NEO4J_DEBUG = False
SEARCH_URL = 'http://127.0.0.1:9200'
OPENRESOLVE_URL = "http://api.openresolve.com/{}/{}"
RIPE_STAT_URL = "https://stat.ripe.net/data/{}/data.json?resource={}"
VIRUSTOTAL_API_SECRET = '<virustotal_api_secret_value>'
VIRUSTOTAL_API_URL = "https://www.virustotal.com/vtapi/v2/{}"
CELERY_QUEUES = [{"name": "providers", "routing_key": "eiq.providers.#"}, {"name":
"analysis", "routing_key": "eiq.analysis.#"}, {"name": "integrations", "routing_key":
"eiq.integrations.#"}, {"name": "enrichers", "routing_key": "eiq.enrichers.#"},
{"name": "utilities", "routing_key": "eiq.utilities.#"}, {"name":
"priority_enrichers", "routing_key": "eiq.priority.enrichers.#"}, {"name":
"priority_providers", "routing_key": "eiq.priority.providers.#"}, {"name":
"priority_utilities", "routing_key": "eiq.priority.utilities.#", {"name":
"reindexing", "routing_key": "eiq.reindexing.#"}]
```

```
# LDAP settings
LDAP_AUTH_ENABLED = False
LDAP_URI = 'ldaps://ldap-server.local'
LDAP_IGNORE_TLS_ERRORS = True

LDAP_BIND_DN = "cn=Manager,dc=ldap,dc=eclecticiq"
LDAP_BIND_PASSWORD = "adminpassword"

LDAP_USERS_FILTER = (
    "ou=Users,dc=ldap,dc=eclecticiq", "(uid={username})")
LDAP_GROUPS_FILTER = (
    "ou=EclecticIQGroups,dc=ldap,dc=eclecticiq",
    "(&(memberUid={username})(objectClass=posixGroup))")
LDAP_ROLES_FILTER = (
    "ou=EclecticIQRoles,dc=ldap,dc=eclecticiq",
    "(&(memberUid={username})(objectClass=posixGroup))")

LDAP_USER_FIRSTNAME_ATTR = 'cn'
LDAP_USER_LASTNAME_ATTR = 'sn'
LDAP_USER_EMAIL_ATTR = 'mail'
LDAP_ROLE_NAME_ATTR = 'cn'
LDAP_GROUP_NAME_ATTR = 'cn'
```

Warning:

Secret key and session token

When you install, update or reinstall the platform, it is a good idea to change the following default values in the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* configuration file:

- `SECRET_KEY = 'secret'`: change the secret key default value to a longer, random one.
- `JWT_EXPIRATION_DELTA = 60 * 60 * 2`: change the expiration time of the user authentication token as needed. The value is measured in seconds.

By default, the token expires 2 hours after successfully signing in. The corresponding session is terminated, and you need to sign back in to the platform. When human interaction is detected — for example, keystrokes or mouse activity — the token is automatically refreshed every 2 minutes. This prevents the system from signing out users who may be working or saving data at that time. Therefore, the default maximum amount of idle time without any human interaction before being automatically signed out equals to *session token validity - 2 minutes*.

- When you set the spool directory for the graph ingestion in the platform configuration file, make sure the `eclecticiq` user can access it in write mode:
  `GRAPH_INGESTION_SPOOL_DIRECTORY = '/opt/eclecticiq/tmp/gi-storage' # 'eclecticiq' user needs write rights to this dir`
- Review the */opt/eclecticiq/etc/eclecticiq/opentaxii.yml* OpenTAXII settings to make sure they reflect your environment.

OpenTAXII configuraton

The */opt/eclecticiq/etc/eclecticiq/opentaxii.yml* file contains the OpenTAXII configuration settings.
The following example serves as a guideline:

```
auth_api:
    class: eiq.opentaxii.auth.PlatformAuthAPI
    parameters: null
domain: <platform_host_name>
hooks: null
logging:
    opentaxii: debug
    root: debug
persistence_api:
    class: eiq.opentaxii.persistence.PlatformPersistenceAPI
    parameters: null
```

Configure LDAP authentication

- To configure LDAP authentication, append the following attributes and set them to the appropriate values in the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* platform settings file.
- Ensure you represent platform roles and groups as LDAP groups.
- LDAP role and group names, that is, the values of the `LDAP_ROLE_NAME_ATTR` and `LDAP_GROUP_NAME_ATTR` attributes, should exactly match the corresponding values in the platform.

- `LDAP_AUTH_ENABLED`: Boolean switch to enable/disable LDAP authentication.
  Default value: `False` (LDAP disabled).
- `LDAP_URI`: URI pointing to the designated to LDAP instance.
  Example:

```
LDAP_URI = "ldaps://ldapserver.example.com"
```

- `LDAP_IGNORE_TLS_ERRORS`: Boolean switch to enable/disable ignoring TLS errors like invalid certificates, chain validation issues, and so on.
  Default value: `True` (TLS errors are ignored).

- `LDAP_BIND_DN` and `LDAP_BIND_PASSWORD`: credentials used to create bindings to the LDAP connection,search for users, read groups and roles.
  Example:

```
LDAP_BIND_DN = "cn=Manager,dc=ldap,dc=eclecticiq"
LDAP_BIND_PASSWORD = "adminpassword"
```

- `LDAP_USERS_FILTER`: base and filter values used to search for user's account.
  The `{username}` placeholder value is to be replaced with a user's assigned user name.
  Example:

```
LDAP_USERS_FILTER = ("ou=Users,dc=ldap,dc=eclecticiq", "(uid={username})")
```

- `LDAP_GROUPS_FILTER`: base and filter values used to search for user groups.
  The `{username}` placeholder value is to be replaced with a user's assigned user name.
  Example:

```
LDAP_GROUPS_FILTER = (
    "ou=EclecticIQGroups,dc=ldap,dc=eclecticiq",
    "(&(memberUid={username})(objectClass=posixGroup))")
```

- `LDAP_ROLES_FILTER`: base and filter values used to search for user's groups.
  The `{username}` placeholder value is to be replaced with a user's assigned user name.
  Example:

```
LDAP_ROLES_FILTER = (
    "ou=EclecticIQRoles,dc=ldap,dc=eclecticiq",
    "(&(memberUid={username})(objectClass=posixGroup))")
```

- `LDAP_USER_FIRSTNAME_ATTR`, `LDAP_USER_LASTNAME_ATTR`, and `LDAP_USER_EMAIL_ATTR`: attribute names used to extract user details.
- `LDAP_ROLE_NAME_ATTR` and `LDAP_GROUP_NAME_ATTR`: attribute names used to extract role and group details.

Bootstrap the platform

Summary: As a last step after installation and configuration, bootstrap the platform and third-party products it interacts with, and then launch the platform to access it.

After installing and configuring the EclecticIQ Platform in the previous steps, you can now proceed to bootstrap the platform before launching it.

Load the Supervisor configuration

Supervisor `(http://supervisord.org/)` is a process manager/task runner. It is shipped with the EclecticIQ Platform.
For reference, the **install package is available** `(https://apps.fedoraproject.org/packages/supervisor)` on the **Fedora EPEL repository** `(https://fedoraproject.org/wiki/epel)` .

To check if Supervisor is installed, run the following command(s):

```
$ sudo yum info supervisor
```

Supervisor is shipped with the platform. If for some reason you need to install it, run the following command(s):

```
$ sudo yum install supervisor
```

This installs two components:

- `supervisord`: the daemon
- `supervisorctl`: the command line interface.

Launch Supervisor to load the configuration to start the tasks and processes other platform components depend on:

```
$ sudo service supervisord start
```

To check if the `supervisord` daemon is running, run the following command(s):

```
$ sudo service supervisord status
```

To check the statuses of the tasks managed by Supervisor, run the following command(s):

```
$ supervisorctl status
```

You may need system administrator rights to run some commands.
In this case, prefix `sudo` to the command.

These are examples of `supervisorctl` actions:

By default, Supervisor configuration files are stored here:

- Supervisor configuration file: */etc/supervisord.conf*
- Additional `supervisord` configuration files: */etc/supervisord.d/*
  (These files are also symlinked to */opt/eclecticiq/etc/supervisord.d/*.)

Warning: When you modify or update the Supervisor configuration, you need to run `$ sudo supervisorctl reload` to update it and to reload the up-to-date configuration into the platform.

For further information on `supervisord` and `supervisorctl`, see the **official documentation** `(http://supervisord.org/running.html)`.

Set up the database

If you are installing the platform for the first time, or if it is a fresh install, there is no database yet. To set up the database, you first need to create it, and then you load the default fixtures for the platform.

To create the database schema, run the following command(s):

```
$ /opt/eclecticiq/migrations/create-db.sh
```

To generate the default platform fixtures, run the following command(s):

```
$ /opt/eclecticiq/migrations/load-fixtures.sh
```

Bootstrap Elasticsearch

To bootstrap Elasticsearch, you need to create its index. To do so, run the following command(s):

```
$ /opt/eclecticiq/platform/api/bin/manage synchronize_elasticsearch
```

Reindex Elasticsearch

To make sure you are applying the latest Elasticsearch schema, fully reindex Elasticsearch.
To do so, run the same command you would execute to create the Elasticsearch index.

- Configure the platform settings environment variable by prepending it to the `manage synchronize_elasticsearch` command.
- Run `manage synchronize_elasticsearch` to reindex all Elasticsearch indexes.
  The command runs in the background. In case of an SSH disconnection, the process should keep running normally.

```
$ EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/manage synchronize_elasticsearch > ./reindexing.log
2>&1 &
```

- Reindexing messages are logged to *reindexing.log*. You can open this file to inspect the scheduling of the reindexing workers.
  Example:


```
{
  "avg_time_per_object_ms": 7,
  "chunk_time_ms": 57838,
  "event": "search_reindexing.processed_chunk",
  "level": "info",
  "logger": "eiq.platform.search_index",
  "n_processed": 7980,
  "request_body_size": 3349388,
  "timestamp": "2016-08-05T14:31:27.626829Z",
  "total_avg_time_per_object_ms": 17
}
```


Bootstrap Neo4j

To bootstrap Neo4j, you need to:

- Create the graph schema
- Apply the graph database migrations.

Prerequisites

Before proceeding to bootstrap Neo4j, verify that the following conditions are met:

- Make sure the Neo4j settings in the *platform_settings.py* configuration file are correct, based on your system environment.
  Typical/Default values are:

```
NEO4J_URL: 'http://localhost:7474'
NEO4J_BATCHING_URL = 'http://127.0.0.1:4008'
NEO4J_DEBUG: False
```

`neo4j-batching`	Neo4j graph database batch processing application. It lives on the same server hosting the Neo4j database. It prepares data for ingestion into the Neo4j database

Neo4j should be up and running:

- Check if Neo4j is running by making a cURL call to the URL defined in `NEO4J_URL` in the in the *platform_settings.py* configuration file.
  By default, it is `http://localhost:7474`.
- If Neo4j is not running, restart the service by running the following command(s):

```
$ sudo service neo4j start
```

or:

```
$ sudo supervisorctl start neo4j
```

Create the graph schema

- Before creating the graph schema, stop the `graph-ingestion` and any running `intel-ingestion` tasks:

```
$ sudo supervisorctl stop graph-ingestion
$ sudo supervisorctl stop intel-ingestion:*
```

- Then, run the following command to create the graph schema and to apply the necessary graph database migrations to the Neo4j database:

```
$ /opt/eclecticiq/platform/api/bin/manage do_graph_migrations
```

Load the dashboard

To populate the plaform dashboard, you need to fetch data from Elasticsearch and Kibana. **elasticdump** (`https://github.com/taskrabbit/elasticsearch-dump`) helps you do that.
To load the platform dashboard, run the following command(s):

```
# Run elasticdump to load the platform dashboard
$ elasticdump --input=/opt/eclecticiq/etc/kibana-dashboard.json --
output=http://localhost:9200/-kibana
```

Run a final check

As a last step before launching the platform, it is good practice to check the following points:

- Core processes and services
- Search, indexing and graph
- Availability

Check core processes and services

Nginx

To check if Nginx is running, run the following command(s):

```
$ sudo service nginx status
```

Supervisor

To check if the `supervisord` daemon is running, run the following command(s):

```
$ sudo service supervisord status
```

PostgreSQL

To check if PostgreSQL is running, run the following command(s):

```
$ sudo service postgresql-9.5 status
```

or:

```
$ sudo systemctl list-units | grep -i postgre
```

Check search indexing and graph

Elasticsearch

To check if Elasticsearch is running, run the following command(s):

```
$ sudo service elasticsearch status
```

Neo4j

Check the main connectivity properties to verify that Neo4j can communicate with other components.

Check neo4j-server.properties

- Open the *neo4j-server.properties* configuration file:

```
$ nano /etc/neo4j/neo4j-server.properties
```

- Make sure that the following properties are set to the corresponding values:

```
org.neo4j.server.webserver.address=0.0.0.0
org.neo4j.server.webserver.port=7474
org.neo4j.server.database.location=/<some_dir>/graph.db
dbms.security.auth_enabled=False
```

`graph.db` is the default name of the graph database file.
Make sure the directory where it is stored exists before setting it in this configuration file, and that Neo4j can read and write data to it.

Inspect the amount of allocated memory: Neo4j purrs when you give it plenty of memory to work with.

- Depending on your system resources, allocate at least 4 GB to Neo4j.
- Never set it below 1 GB.

To set the minimum and maximum memory values for Neo4j, do the following:

Check neo4j-wrapper.conf

- Open the *neo4j-wrapper.conf* configuration file:

```
$ nano /etc/neo4j/neo4j-wrapper.conf
```

- Make sure that the following properties are set to the corresponding values:

```
wrapper.java.initmemory=1024
wrapper.java.maxmemory=4096
```

- The recommended value for `wrapper.java.initmemory` is at least 4096 MB or more, if possible. The minimum value should never be lower than 1024 MB.
- The recommended value for `wrapper.java.maxmemory` is at least 4096 MB or more, if possible. This value should never be lower than `wrapper.java.initmemory`.

Check neo4j.properties

Set the size of the page cache to `5G`:

- Open the *neo4j.properties* property file:

```
$ nano /etc/neo4j/neo4j.properties
```

- Make sure the `dbms.pagecache.memory` property is set to `5G`:

```
dbms.pagecache.memory=5G
```

If this parameter is not configured, Neo4j defaults to using 50% of the available system RAM for page caching.

Check availability

Make sure Elasticsearch and Neo4j are available by making cURL calls to the corresponding endpoints:

```
# Check Elasticsearch availability
$ curl localhost:9200

# Check Neo4j availability
$ curl localhost:7474
```

If Neo4j is not running, restart the service by running the following command(s):

```
$ sudo service neo4j start
```

or:

```
$ sudo supervisorctl start neo4j
```

Reload the Supervisor configuration

To reload the Supervisor configuration and to restart all Supervisor-managed processes run the following command(s):

```
$ sudo supervisorctl reload
```

To check the statuses of the tasks managed by Supervisor, run the following command(s):

```
$ supervisorctl status
```

The response should return `RUNNING` for all relevant tasks to confirm that all Supervisor tasks are being executed normally.

The following example serves as a guideline:

```
graph-ingestion                  RUNNING   pid 3443,  uptime 1:10:56
intel-ingestion:0                RUNNING   pid 3323,  uptime 1:11:19
intel-ingestion:1                RUNNING   pid 3336,  uptime 1:11:14
intel-ingestion:2                RUNNING   pid 3363,  uptime 1:11:09
intel-ingestion:3                RUNNING   pid 3372,  uptime 1:11:04
kibana                           RUNNING   pid 13837, uptime 9 days, 0:38:17
neo4j-batching                   RUNNING   pid 3590,  uptime 1:10:45
opentaxii                        RUNNING   pid 3662,  uptime 1:10:39
platform-api                     RUNNING   pid 2999,  uptime 1:12:47
search-ingestion                 RUNNING   pid 3513,  uptime 1:10:49
task:beat                        RUNNING   pid 3102,  uptime 1:12:39
task:enrichers                   RUNNING   pid 3110,  uptime 1:12:26
task:integrations                RUNNING   pid 3135,  uptime 1:12:13
task:providers                   RUNNING   pid 3204,  uptime 1:12:01
task:reindexing                  RUNNING   pid 3223,  uptime 1:11:48
task:utilities                   RUNNING   pid 3242,  uptime 1:11:35
```

Warning: When you modify or update the Supervisor configuration, you need to run `$ sudo supervisorctl reload` to update it and to reload the up-to-date configuration into the platform.

Access the platform

To access the platform, do the following:

- Launch a web browser (recommended: Google Chrome).
- Go to the configured platform address, for example: `https://platform.host`
- On the login page, enter the appropriate credentials.

Warning: The browser may display an untrusted connection warning: add it as an exception, and then proceed to the platform.

Upgrade the platform

Summary: When a new platform release is available, you can upgrade your existing installation to benefit from the latest features and enhancements.

When you download a new RPM package containing a newer platform release than the currently installed one on your system, you can upgrade your platform installation to the latest available public version.

The upgrade procedure requires some housekeeping; once you are done, you can access new features, and you can enjoy the improvements we introduce in the product on a regular basis.

The upgrade procedure consists of the following steps:

Before the upgrade

- Exit the platform
- Back up your data
- Check the prerequisites
- Remove deprecated packages

During the upgrade

- Install the new package

After the upgrade

- Check the configuration
- Check third-party configurations
- Migrate the database
- Check component versions
- Run the fixtures
- Run a final check
- Reload the Supervisor configuration
- Access the platform

Exit the platform

Sign out of the platform:

- Click the active user profile image on the top-right corner of the screen.
- From the drop-down menu select **Sign out**.
- You are signed out.

Back up your data

Warning: Before proceeding to upgrade the platform or any of its third-party components, always back up your data.

Check the prerequisites

Before installing the new platform release, check the prerequisites, and verify that the following conditions are met:

- The required repositories are already installed on the target system.
- All required third-party components are already installed on the target system, and their versions match the recommended version information provided.
- To perform several tasks during the procedure, you need root-level permissions.
  To obtain administration rights, run the following command(s):

```
$ sudo su -
```

Remove deprecated packages

Before installing the new platform release, make sure you remove any deprecated packages.
To remove deprecated packages, run the following command(s):

```
$ sudo yum remove <package_name>
```

or:

```
$ sudo rpm -e <package_name>
```

In case uninstalling a deprecated package fails because of dependency-related errors, include the `--nodeps` **option** `(http://www.rpm.org/max-rpm-snapshot/s1-rpm-erase-additional-options.html#s2-rpm-erase-nodeps-option)`:

```
$ sudo rpm -e --nodeps <package_name>
```

The following packages are *deprecated*: *remove* them before upgrading.

Before upgrading to this version…	…remove this RPM package
0.17	`eclecticiq-intel-search-0.16.0-1.x86_64`
1.11.0	`platform-opentaxii`

Upgrade to release 1.11.0
To verify that `platform-opentaxii` has been removed, carry out the following steps after completing the upgrade:

- Check that the `/opt/eclecticiq/platform/opentaxii` directory does not exist in the system hosting release 1.11.0.
- If the directory exists, delete it.
- Restart Supervisor-managed processes.
- To quickly verify that the TAXII interface is up and running, execute the following **Cabby CLI command** (`http://cabby.readthedocs.io/en/latest/user.html#using-cabby-as-a-command-line-tool`) to fetch the TAXII services the platform advertises at `https://<platform_host>/taxii/discovery`:

```
$ taxii-discovery --path https://<platform_host>/taxii/discovery --verify no
```

Install the new package

You can now proceed to the upgrade step:

- You will receive a download link. Follow it, download the *eclecticiq-platform-x.x.x.zip .zip* archive, and save it to the target location on the host system. For example save it to the root home directory.
- Install the new RPM package following the standard installation procedure.

Before starting the installation, a script carries out a version check to detect any previously installed platform versions.

It is possible to upgrade the platform only sequentially, it is not possible to skip versions.
Therefore, if you want to upgrade from release 1.10.0 to 1.14.0, you need to upgrade step by step by installing all intermediate releases until 1.14.0.
Example:

```
# Upgrade from 1.10.0 to 1.10.1
$ sudo yum install -y eclecticiq-platform-1.10.1

# Upgrade from 1.10.1 to 1.11.0
$ sudo yum install -y eclecticiq-platform-1.11.0

# Upgrade from 1.11.0 to 1.12.0
$ sudo yum install -y eclecticiq-platform-1.12.0

# Upgrade from 1.12.0 to 1.13.0
$ sudo yum install -y eclecticiq-platform-1.13.0

# Upgrade from 1.13.0 to 1.14.0
$ sudo yum install -y eclecticiq-platform-1.14.0
```

If the currently installed version details are the same as the corresponding information in the upgrade version, or if the former does not immediately precedes the latter, the upgrade procedure is aborted, and an error message is displayed:

```
You currently have version ${INSTALLED_VERSION}, but you need to have ${PREV_VERSION}
installed in order to upgrade ${PACKAGE_NAME}.

exit 99
```

`${INSTALLED_VERSION}`	The currently installed version on your system. The upgrade fails because this version is too old.
`${PREV_VERSION}`	Before starting the upgrade procedure again, install this platform version on your system. The upgrade procedure works only when the old and the new versions are contiguous in sequence.
`exit 99`	The exit code accompanying the error message.

After a successful platform installation, the following message is displayed to inform you that you need to carry out a few post-installation tasks to complete the upgrade procedure:

```
You are almost there, just a few more steps before you complete the EclecticIQ
Platform upgrade procedure.
Refer to the upgrade section in the EclecticIQ Platform RPM installation and
configuration guide to learn what you should do next.
```

Just follow the next sections in this document to proceed to the configuration and bootstrapping steps.

Check the configuration

After installing the platform, browse to */opt/eclecticiq/etc/eclecticiq/*. Configuration files are stored here. You can find both the new/latest configuration files, as well as the ones belonging to the previous version of the platform.

platform_settings.py	The main configuration file for the platform
opentaxii.yml	The **OpenTAXII** `(http://opentaxii.readthedocs.org/)` configuration file

Verify that the platform configuration files reflect the new, upgraded environment. For example Neo4j has been upgraded from version 2.24 to 2.3.1: make sure the environment described in the upgraded platform configuration file refers to the higher version number.

You may need to carry out this task manually. In this case, you can diff the files with a tool like **Meld** `(http://meldmerge.org/)`.

Check third-party configurations

After checking the platform configuration to make sure it correctly describes the upgraded environment, do the same with third-party product configurations.

You may need to carry out this task manually. In this case, you can diff the files with a tool like **Meld** `(http://meldmerge.org/)`.

Migrate the database

At this point, you verified that the platform upgrade was installed successfully, and you reviewed the configurations to make sure they correctly reflect the upgraded platform environment.

Time to take care of the database migration.

If you are upgrading the platform to a newer release, you need to migrate the existing database to the new platform release.
To perform the database migration, run the following script:

```
$ /opt/eclecticiq/migrations/db-migration.sh
```

Reindex Elasticsearch

To make sure you are applying the latest Elasticsearch schema, fully reindex Elasticsearch.
To do so, run the same command you would execute to create the Elasticsearch index.

- Configure the platform settings environment variable by prepending it to the `manage synchronize_elasticsearch` command.
- Run `manage synchronize_elasticsearch` to reindex all Elasticsearch indexes.
  The command runs in the background. In case of an SSH disconnection, the process should keep running normally.

```
$ EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/manage synchronize_elasticsearch > ./reindexing.log
2>&1 &
```

- Reindexing messages are logged to *reindexing.log*. You can open this file to inspect the scheduling of the reindexing workers.
  Example:

```
{
  "avg_time_per_object_ms": 7,
  "chunk_time_ms": 57838,
  "event": "search_reindexing.processed_chunk",
  "level": "info",
  "logger": "eiq.platform.search_index",
  "n_processed": 7980,
  "request_body_size": 3349388,
  "timestamp": "2016-08-05T14:31:27.626829Z",
  "total_avg_time_per_object_ms": 17
}
```

Migrate the graph database

Make sure Neo4j is up and running before applying the graph database migrations.

- Before creating the graph schema, stop the `graph-ingestion` and any running `intel-ingestion` tasks:

```
$ sudo supervisorctl stop graph-ingestion
$ sudo supervisorctl stop intel-ingestion:*
```

- Then, run the following command to create the graph schema and to apply the necessary graph database migrations to the Neo4j database:

```
$ /opt/eclecticiq/platform/api/bin/manage do_graph_migrations
```

Check component versions

After migrating the database, check the versions of the upgraded components to verify that they are the correct ones.
Authenticate with the platform API to receive a bearer token, then pass it to the `/api/versions` API endpoint to obtain version information:

```
$ curl -X GET
       -v
       --insecure
       -i
       -H "Content-Type: application/json"
       -H "Accept: application/json"
       -H "Authorization: Bearer <token>"
       https://platform.host/api/versions
```

The JSON response contains version information about the core platform components:

```
{
  "data": {

    "opentaxii": {
      "source": "devpi",
      "tag": "latest",
      "version": "1.12.0"
    },

    "opentaxii-platform-apis": {
      "branch": "master",
      "source": "github",
      "summary": "release/1.12.0",
      "tag": "latest",
      "version": "bc0478b3d10da0a3583563814e05d53116cfaba4"
    },

    "platform-api": {
      "branch": "master",
      "source": "github",
      "summary": "release/1.12.0",
      "tag": "latest",
      "version": "1b7d297394d716da79f9310f377ef72835b8427e"
    },

    "platform-database": {
      "current": "123a456bcd7",
      "head": "123a456bcd7 (head)",
      /*
        Allowed status values:
          - Up to date
          - Outdated
          - Head not available (when no manifest file is found)
      */
      "status": "Up to date"
    },

    "platform-ui": {
      "source": "artifactory",
      "tag": "latest",
      "version": "release/1.12.0"
    }

  }
}
```


This version information matches the corresponding details in the manifest files.
Manifest files are stored in the following directory:

```
$ cd /opt/eclecticiq/manifests/
```

The directory contains these manifest files:

```
opentaxii.mf
opentaxii-platform-apis.mf
platform-api.mf
platform-database.mf
platform-ui.mf
```

As for the (migrated) database, the current database version needs to match the one in the `platform-database.mf` manifest file:

- Request the current database version:

```
$ cd /opt/eclecticiq/migrations/ &&
EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/manage alembic current
```

Example:

```
123a456bcd7 (head)
```

- Go to the manifest directory:

```
$ cd /opt/eclecticiq/manifests/
```

- Open the *platform-database.mf* manifest file:

```
$ nano platform-database.mf
```

- Verify that the database head hash in the manifest matches the returned current database version:

```
head: 123a456bcd7 (head)
```

Run the fixtures

To generate the default platform fixtures, run the following command(s):

```
$ /opt/eclecticiq/migrations/load-fixtures.sh
```

Run a final check

As a last step before launching the platform, it is good practice to check the following points:

- Core processes and services
- Search, indexing and graph
- Availability

Check core processes and services

Nginx

To check if Nginx is running, run the following command(s):

```
$ sudo service nginx status
```

Supervisor

To check if the `supervisord` daemon is running, run the following command(s):

```
$ sudo service supervisord status
```

PostgreSQL

To check if PostgreSQL is running, run the following command(s):

```
$ sudo service postgresql-9.5 status
```

or:

```
$ sudo systemctl list-units | grep -i postgre
```

Check search indexing and graph

Elasticsearch

To check if Elasticsearch is running, run the following command(s):

```
$ sudo service elasticsearch status
```

Neo4j

Check the main connectivity properties to verify that Neo4j can communicate with other components.

Check neo4j-server.properties

- Open the *neo4j-server.properties* configuration file:

```
$ nano /etc/neo4j/neo4j-server.properties
```

- Make sure that the following properties are set to the corresponding values:

```
org.neo4j.server.webserver.address=0.0.0.0
org.neo4j.server.webserver.port=7474
org.neo4j.server.database.location=/<some_dir>/graph.db
dbms.security.auth_enabled=False
```

`graph.db` is the default name of the graph database file.
Make sure the directory where it is stored exists before setting it in this configuration file, and that Neo4j can read and write data to it.

Inspect the amount of allocated memory: Neo4j purrs when you give it plenty of memory to work with.

- Depending on your system resources, allocate at least 4 GB to Neo4j.
- Never set it below 1 GB.

To set the minimum and maximum memory values for Neo4j, do the following:

Check neo4j-wrapper.conf

- Open the *neo4j-wrapper.conf* configuration file:

```
$ nano /etc/neo4j/neo4j-wrapper.conf
```

- Make sure that the following properties are set to the corresponding values:

```
wrapper.java.initmemory=1024
wrapper.java.maxmemory=4096
```

- The recommended value for `wrapper.java.initmemory` is at least 4096 MB or more, if possible.
  The minimum value should never be lower than 1024 MB.
- The recommended value for `wrapper.java.maxmemory` is at least 4096 MB or more, if possible.
  This value should never be lower than `wrapper.java.initmemory`.

Check neo4j.properties

Set the size of the page cache to `5G`:

- Open the *neo4j.properties* property file:

```
$ nano /etc/neo4j/neo4j.properties
```

- Make sure the `dbms.pagecache.memory` property is set to `5G`:

```
dbms.pagecache.memory=5G
```

If this parameter is not configured, Neo4j defaults to using 50% of the available system RAM for page caching.

Check availability

Make sure Elasticsearch and Neo4j are available by making cURL calls to the corresponding endpoints:

```
# Check Elasticsearch availability
$ curl localhost:9200

# Check Neo4j availability
$ curl localhost:7474
```

If Neo4j is not running, restart the service by running the following command(s):

```
$ sudo service neo4j start
```

or:

```
$ sudo supervisorctl start neo4j
```

Reload the Supervisor configuration

To reload the Supervisor configuration and to restart all Supervisor-managed processes run the following command(s):

```
$ sudo supervisorctl reload
```

To check the statuses of the tasks managed by Supervisor, run the following command(s):

```
$ supervisorctl status
```

The response should return `RUNNING` for all relevant tasks to confirm that all Supervisor tasks are being executed normally.

The following example serves as a guideline:

```
graph-ingestion                  RUNNING   pid 3443,  uptime 1:10:56
intel-ingestion:0                RUNNING   pid 3323,  uptime 1:11:19
intel-ingestion:1                RUNNING   pid 3336,  uptime 1:11:14
intel-ingestion:2                RUNNING   pid 3363,  uptime 1:11:09
intel-ingestion:3                RUNNING   pid 3372,  uptime 1:11:04
kibana                           RUNNING   pid 13837, uptime 9 days, 0:38:17
neo4j-batching                   RUNNING   pid 3590,  uptime 1:10:45
opentaxii                        RUNNING   pid 3662,  uptime 1:10:39
platform-api                     RUNNING   pid 2999,  uptime 1:12:47
search-ingestion                 RUNNING   pid 3513,  uptime 1:10:49
task:beat                        RUNNING   pid 3102,  uptime 1:12:39
task:enrichers                   RUNNING   pid 3110,  uptime 1:12:26
task:integrations                RUNNING   pid 3135,  uptime 1:12:13
task:providers                   RUNNING   pid 3204,  uptime 1:12:01
task:reindexing                  RUNNING   pid 3223,  uptime 1:11:48
task:utilities                   RUNNING   pid 3242,  uptime 1:11:35
```

Access the platform

To access the platform, do the following:

- Launch a web browser (recommended: Google Chrome).
- Go to the configured platform address, for example: `https://platform.host`
- On the login page, enter the appropriate credentials.

Warning: The browser may display an untrusted connection warning: add it as an exception, and then proceed to the platform.

Backup guidelines

Summary: Back up platform data to restore it when you upgrade or reinstall the platform, and as a disaster recovery mitigation strategy.

As a best practice, we recommend you implement a backup strategy for platform data. Backups come in handy in situations like:

- Platform upgrade to a newer release
- Platform reinstallation
- Disaster recovery.

Before starting a platform backup, verify the following points:

- No users are signed in to the platform
- No read/write activity is in progress
- The platform is not running.

The core data you should include in a platform backup are:

- Configuration files
- Databases.

The exact steps to back up platform data may vary, depending on your sepcific environment hardware, configuration, and setup. Therefore, consider the following as a set of generic guidelines on backing up platform data.

Platform configuration

Back up the platform configuration files to restore the platform setup at a later time.
To get a list with the platform configuration files, run the following command(s):

```
# Returns only core config files
$ find /opt/eclecticiq/etc/eclecticiq/ -type f

# Returns platform and third-party components config files
$ find /opt/eclecticiq/etc/ -type f

# Returns third-party config files to use for reference as boilerplates
$ find /opt/eclecticiq/etc-extras/ -type f
```

The response returns the configuration files to include in the backup:

```
# Core platform settings
```

```
/opt/eclecticiq/etc/eclecticiq/opentaxii.yml
/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/etc/eclecticiq/proxy_url

# Dashboard settings
/opt/eclecticiq/etc/kibana-dashboard.json

# Logstash log aggregation settings
/opt/eclecticiq/etc/logstash/conf.d/filters.conf
/opt/eclecticiq/etc/logstash/conf.d/input.conf
/opt/eclecticiq/etc/logstash/conf.d/neo4j-batching.conf
/opt/eclecticiq/etc/logstash/conf.d/opentaxii.conf
/opt/eclecticiq/etc/logstash/conf.d/output.conf
/opt/eclecticiq/etc/logstash/conf.d/platform-api.conf
/opt/eclecticiq/etc/logstash/conf.d/platform-ui.conf

# Logstash heap size
/opt/eclecticiq/etc/sysconfig/logstash

# Web server, proxy and port settings
/opt/eclecticiq/etc/nginx/conf.d/platform.conf

# Platform components ini file settings: ingestion, search, graph, taxii, api, tasks
/opt/eclecticiq/etc/supervisord.d/graph-ingestion.ini
/opt/eclecticiq/etc/supervisord.d/intel-ingestion.ini
/opt/eclecticiq/etc/supervisord.d/neo4j-batching.ini
/opt/eclecticiq/etc/supervisord.d/opentaxii.ini
/opt/eclecticiq/etc/supervisord.d/platform-api.ini
/opt/eclecticiq/etc/supervisord.d/search-ingestion.ini
/opt/eclecticiq/etc/supervisord.d/task-workers.ini

# Kibana config file
/opt/eclecticiq/etc-extras/kibana.yml

# Redis message broker config file
/opt/eclecticiq/etc-extras/redis.conf

# Elasticsearch config files
/opt/eclecticiq/etc-extras/elasticsearch/elasticsearch.yml
/opt/eclecticiq/etc-extras/elasticsearch/logging.yml

# Neo4j config files
/opt/eclecticiq/etc-extras/neo4j/neo4j.properties
/opt/eclecticiq/etc-extras/neo4j/neo4j-server.properties
/opt/eclecticiq/etc-extras/neo4j/neo4j-wrapper.conf

# Nginx web server config file
/opt/eclecticiq/etc-extras/nginx/nginx.conf

# Postfix email server config file
/opt/eclecticiq/etc-extras/postfix/main.cf

# Kibana and Neo4j ini files
/opt/eclecticiq/etc-extras/supervisord/kibana.ini
/opt/eclecticiq/etc-extras/supervisord/neo4j-console.ini
```

Platform databases

The EclecticIQ Platform uses the following databases:

Database	Description
PostgreSQL (http://www.postgresql.org/docs/manuals/)	Intel database. It stores all the information about entities, extracts, relationships, taxonomy, and so on.
Elasticsearch (https://www.elastic.co/guide/index.html)	Indexing and search database. It stores document data and search queries as JSON.
Neo4j (http://neo4j.com/docs/)	Graph database. It stores node, edge, and property information to build and represent data relationships.

It is best to include all databases in your backup strategy. If for any reason it is not possible, make sure that at least the PostgreSQL database is backed up on a regular basis.

Backup guidelines

Back up the PostgreSQL database

You can back up a PostgreSQL database in several ways:

- SQL dump (recommended)
- File system level backup
- Continuous archiving

SQL dump

The quickest way to backup a PostgreSQL database is to perform an **SQL dump** (http://www.postgresql.org/docs/current/static/backup-dump.html). To generate an SQL dump of the database, run the **pg_dump** (http://www.postgresql.org/docs/current/static/app-pgdump.html) or the **pg_dumpall** (http://www.postgresql.org/docs/current/static/app-pg-dumpall.html) command.

To create an *.sql* dump file, run the following command(s):

```
$ sudo -u postgres /usr/pgsql-9.5/bin/pg_dumpall > /output/db_dump.sql
```

To restore a backed up database from an *.sql* dump file, run the following command(s):

```
$ psql -U postgres < ./db_dump.sql
```

File system level backup

File system level backup `( http://www.postgresql.org/docs/current/static/backup-file.html)` creates backup copies of the PostgreSQL file used to store the database data corpus. The files to back up are stored in the **database cluster** `(http://www.postgresql.org/docs/current/static/creating-cluster.html)` specified with the **`initdb`** `(http://www.postgresql.org/docs/current/static/app-initdb.html)` command during database storage location inizialization.
This approach requires database downtime.

To back up a database by archiving the files PostgreSQL uses to store data in the database, run the following command(s):

- Go to the PostgreSQL data directory, typically *../pgsql/n<version>/data/*:

```
$ cd /var/lib/pgsql/9.5/data/
```

- Create a *.tar* archive containing the entire content of the `data` directory:

```
$ tar -cvf db_backup.tar *
```

To restore a database by copying the files PostgreSQL uses to store data in the database, run the following command(s):

- Copy the *.tar* archive you just created to the target environment.
- In the target environment, delete any content in the *data* directory:

```
$ rm -rf /var/lib/pgsql/9.5/data/*
```

- Go to the PostgreSQL data directory where you want to restore the database data, typically *../pgsql/n<version>/data/*:

```
$ cd /var/lib/pgsql/9.5/data/
```

- Decompress the *.tar* archive:

```
$ tar -xvf db_backup.tar
```

Continuous archiving

Continuous archiving (http://www.postgresql.org/docs/current/static/continuous-archiving.html) allows you to back up and restore a snapshot of the database in the state it was at a given point in time. It combines file system level backup with write-ahead logging (WAL).

These are the main steps you need to carry out to set up this backup strategy:

- Configure the **write-ahead log** (http://www.postgresql.org/docs/current/static/wal-configuration.html) behavior. Usually, a section in the *postgresql.conf* file contains the WAL parameters you need to define.
  For example you would typically set **`wal_level`** (http://www.postgresql.org/docs/current/static/runtime-config-wal.html#guc-wal-level) to `archive`, **`archive_mode`** (http://www.postgresql.org/docs/current/static/runtime-config-wal.html#guc-archive-mode) to `on`, and you may wish to set an **`archive_command`** (http://www.postgresql.org/docs/current/static/runtime-config-wal.html#guc-archive-command), as well as an **`archive_timeout`** (http://www.postgresql.org/docs/current/static/runtime-config-wal.html#guc-archive-timeout).
- Perform a **base backup** (http://www.postgresql.org/docs/current/static/continuous-archiving.html#backup-base-backup) by running **`pg_basebackup`** (http://www.postgresql.org/docs/current/static/app-pgbasebackup.html).
- As an alternative, you can manage backups with **pgbarman** (http://www.pgbarman.org/), an open source tool you can **download here** (https://sourceforge.net/projects/pgbarman/).

Back up the Elasticsearch database

The Elasticsearch official documentation includes sections with explanations of some **key concepts** (https://www.elastic.co/guide/en/elasticsearch/reference/current/_basic_concepts.html), as well as step-by-step tutorials on the following topics:

- **Back up an Elasticsearch database** (https://www.elastic.co/guide/en/elasticsearch/reference/current/backup.html)
- **Back up an Elasticsearch cluster** (https://www.elastic.co/guide/en/elasticsearch/guide/current/backing-up-your-cluster.html)
- Use the **`snapshot` API** (https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-snapshots.html) to create snapshots of an index or a whole cluster.

Back up the Neo4j database

The Neo4j official documentation includes sections with explanations of **backup** `(http://neo4j.com/docs/stable/operations-backup.html)` concepts and procedures:

- **Configure** `(http://neo4j.com/docs/stable/backup-introduction.html)` Neo4j to enable backups
- **Back up** `(http://neo4j.com/docs/stable/backup-performing.html)` the Neo4j database
- Use `neo4j-backup`, the **Neo4j backup tool** `(http://neo4j.com/docs/stable/re04.html)` (shipped with the Neo4j Enterprise edition only).

Neo4j Community edition does not include a backup tool.
The following examples provide simple suggestions to manually start a backup operation.

You may need system administrator rights to run some commands.
In this case, prefix `sudo` to the command.

- Before starting backing up data, stop Neo4j.
  Run the following command(s):

```
$ sudo service neo4j stop
```

- Make sure Neo4j has stopped.
  Run the following command(s):

```
$ sudo service neo4j status
```

- Once Neo4j is stopped, browse to the Neo4j data directory and compress the *graph.db* directory.
  Run the following command(s):

```
$ cd <neo4j_data_dir>
$ tar -cfvz graph.db.tar.gz graph.db/
```

- Copy the compressed file to a different directory.

The Neo4j data location is defined in the */etc/neo4j/neo4j-server.properties* **configuration file** `(http://neo4j.com/docs/stable/server-configuration.html)` as follows:

```
org.neo4j.server.database.location=data/graph.db
```

Data recovery

To restore backed up data, follow the standard recommendations and procedures for PostgreSQL, Elasticsearch, and Neo4j:

- **Back up and restore PostgreSQL data** (https://www.postgresql.org/docs/current/static/backup.html)
- **Restore a PostgreSQL data dump** (https://www.postgresql.org/docs/current/static/backup-dump.html#backup-dump-restore)
- **Restore PostgreSQL data using a continuous archive backup** (https://www.postgresql.org/docs/current/static/continuous-archiving.html#backup-pitr-recovery)
- **Restore PostgreSQL using pg_restore** (http://postgresguide.com/utilities/backup-restore.html)
- **Restore Elasticsearch data with snapshot and restore** (https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-snapshots.html)
- **Restore a Neo4j database backup** (https://neo4j.com/docs/operations-manual/current/backup/#backup-restoring)

Check for failed packages

During a data restore operation, you may wish to check if the scenario that created the need for a data backup restore also caused some packages to be partially or incorrectly ingested into the platform.

The blobs associated with problematic packages are not marked a successful. To retrieve a list with these packages, execute the following SQL query against the PostgreSQL database:

```
$ SELECT id, processing_status FROM blob WHERE processing_status NOT IN ('success', 'pending');
```

The query returns all packages whose status is not `success` or `pending`, as well as additional information on the packages, when available. Examine the response to evaluate whether you want to try reingesting the packages again.