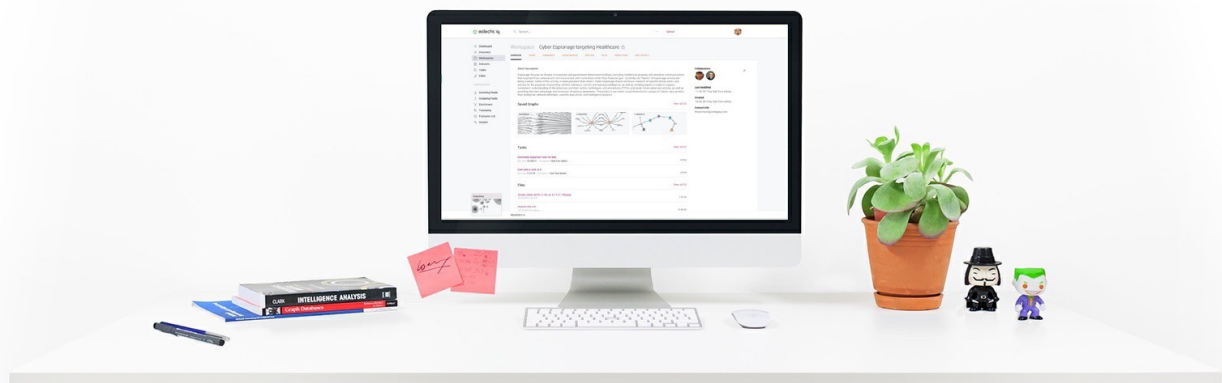




Install and configure EclecticIQ Platform

Install and config guide for system administrators

Last generated: May 26, 2017



©2017 EclecticIQ

All rights reserved. No part of this document may be reproduced in any form or by any electronic or mechanical means, including information storage and retrieval systems, without written permission from the author, except in the case of a reviewer, who may quote brief passages embodied in critical articles or in a review.

Trademarked names appear throughout this book. Rather than use a trademark symbol with every occurrence of a trademarked name, names are used in an editorial fashion, with no intention of infringement of the respective owner's trademark.

The information in this book is distributed on an "as is" basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author nor the publisher shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this book.

©2017 by EclecticIQ BV. All rights reserved.
Last generated on May 26, 2017

Table of contents

Table of contents	2
RPM installation and configuration guide	7
Scope	7
Goal	7
Audience	7
Feedback	8
Before you start	9
About EclecticlQ Platform	9
Hardware requirements	10
Single box	10
Scaling out	11
Software requirements	11
Credentials and host name	11
Operating systems	12
Third-party products	12
Install the dependencies	15
Add the repositories	15
Install the core dependencies	15
EPEL	15
IUS repository	16
Java SE Development Kit	16
Install third-party components	17
Nginx	17
PostgreSQL	18
Configure PostgreSQL (1 of 2)	18
Create the database	19
Set the database authentication method	19
Restart the database service	20
Redis	20
ELK stack	20
Elasticsearch	20
Autostart on boot	21
Install the Elasticsearch plugins	21
Logstash	22
Kibana	22
Neo4j	23
poppler-utils	23
Install via an RPM package	25
Install from a downloaded archive	25
Install from the YUM repository	26
Create the YUM repository configuration	26
Run the YUM install	27
Version check	27
Check SELinux	29
Config and log files	33
Configuration, log and manifest files	33
Configuration files	33
Log files	37
Manifest files	41
Default users	43
Configure the platform	45
Configure PostgreSQL (2 of 2)	45
Add the database to the platform settings	45
Restart the database service	46

Configure Nginx	46
Configure Redis	49
Configure Elasticsearch	51
Configure Logstash	54
Configure Kibana	54
Configure Neo4j	55
Update the platform settings	57
Secret key and session token	59
Configure OpenTAXII	59
Configure LDAP authentication	60
Configure LDAP to work with AD	63
Example	63
Bootstrap the platform	66
Load the Supervisor configuration	66
Set up the database	67
Bootstrap Elasticsearch	68
Reindex Elasticsearch	68
Bootstrap Neo4j	69
Prerequisites	69
Create the graph schema	69
Load the dashboard	70
Run a final check	70
Check core processes and services	70
Check search indexing and graph	71
Check availability	72
Reload the Supervisor configuration	73
Access the platform	74
Upgrade the platform	75
Exit the platform	76
Back up your data	76
Shut down the platform	76
Check the prerequisites	78
Remove deprecated packages	78
Install the new package	79
Check the configuration	81
Check third-party configurations	81
Install poppler-utils	81
Migrate the database	82
Reindex Elasticsearch	82
Migrate the graph database	83
Check component versions	83
Run the fixtures	85
Run a final check	86
Check core processes and services	86
Check search indexing and graph	86
Check availability	88
Reload the Supervisor configuration	88
Access the platform	89
Backup guidelines	90
Platform configuration	90
Platform databases	92
Backup guidelines	92
Back up the PostgreSQL database	92
SQL dump	92
File system level backup	93
Continuous archiving	94

Back up the Elasticsearch database	94
Back up the Neo4j database	95
Data recovery	96
Check for failed packages	96
How to install the platform via an RPM package	97
Scope	97
Goal	97
Audience	97
Prerequisites	97
Credentials and host name	97
Operating systems	98
Create the virtual server instance	98
Configure the virtual server instance	99
Add the repositories	100
Install the core dependencies	100
EPEL	101
IUS repository	101
Java SE Development Kit	101
Install third-party components	102
Nginx	102
PostgreSQL	103
Configure PostgreSQL (1 of 2)	103
Create the database	104
Set the database authentication method	104
Restart the database service	105
Redis	105
ELK stack	105
Elasticsearch	105
Autostart on boot	106
Install the Elasticsearch plugins	106
Logstash	107
Kibana	107
Neo4j	108
Install poppler-utils	108
Set up the EclecticIQ repository	109
Install the platform	109
Configure the platform	110
Third-party configuration files	110
Configure PostgreSQL (2 of 2)	111
Add the database to the platform settings	111
Restart the database service	112
Configure Nginx	112
Configure Redis	115
Configure Elasticsearch	117
Configure Logstash	120
Check the event pipeline	122
Test the configuration	124
Configure Kibana	124
Configure Neo4j	127
Update the platform settings	129
Bootstrap the platform	131
Set up the database	132
Bootstrap Elasticsearch	132
Bootstrap Neo4j	133
Create the graph schema	134
Load the dashboard	134

Reload the Supervisor configuration	134
Access the platform	135
How to check system health	136
Check system health via the GUI	136
Check system health via the API	139
API endpoint	139
Status request	140
Status response	140
How to monitor the platform	144
Prerequisites	144
Scope	144
Goal	144
Audience	145
Tools	145
Core components	145
Monitoring	146
Monitor components with Supervisor	146
Supervisor actions	151
Monitor components with systemd	152
Monitor processes	154
Monitor ingestion queues with Redis	154
Monitor running tasks with Celery	154
Check the Flower database size	156
Whoa! This section is outdated!	156
How to setup Nginx client certificate verification	157
Enable client certificate verification in Nginx	157
Install a client certificate in Google Chrome	157
How to configure a different database in OpenTAXII	159
How to enable audit logging in Kibana	161
Enable audit logging	161
View audit logs	161
View audit logs in Kibana	162
View audit logs in the web interface	163
How to address logging issues in Kibana	165
Prerequisites	165
The ELK stack	165
Check Kibana and Logstash configurations to address logging issues	165
Access Kibana	166
Check the basics	166
Create an index	166
Check Kibana configuration	167
Check Logstash	168
Check Logstash configuration	168
Check platform-api.conf	169
Check elasticsearch.yml	170
Check Logstash event pipeline	171
Check input.conf	171
Check filters.conf	172
Check output.conf	173
Test Logstash configuration	173
Go back to Kibana	173
Adjust the update interval	174
Configure output to syslog	174
How to search logs for issues in Kibana	175
Search Kibana to retrieve log data about issues	175
Access Kibana	175

Adjust the update interval	176
Search by level	176
Search by tag	177
Search using Boolean operators	177
How to reindex Elasticsearch	179
Reindex Elasticsearch	179
Fully reindex Elasticsearch	179
How to reindex the graph database	181
How to back up and restore a PostgreSQL database	183
Back up the PostgreSQL database	183
Restore the backup	183
Check, check, check	184
Check the PostgreSQL configuration	184
Check the platform configuration	185
Check the PostgreSQL service	186
Check for failed packages	186
How to shut down the platform	187

RPM installation and configuration guide

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

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 ingesting, consolidating, analyzing, integrating, and collaborating on intelligence 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/ <i>rpm</i> install	
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
Drive allocation (GB)	10	20	Root (EclecticIQ Platform + Redis)
	10	20	Log data storage

HW area	Minimum	Recommended	Notes
	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

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.

Third-party products

Third-party software includes required dependencies for EclecticIQ Platform to operate correctly.



Make sure that the following software products are *already installed* on the target system *before* installing the platform.
During installation, the platform checks for these dependencies. If they are missing, the installation procedure aborts.

Third-party product	Version	Reference
Oracle Java JDK	1.8.0	Oracle Java download page (http://www.oracle.com/technetwork/java/javase/downloads/index.htm)
PostgreSQL	9.5	PostgreSQL web site (https://yum.postgresql.org/repos/packages.php)
Redis	3.2.3-1.el7	Redis web site (http://redis.io/download)
Nginx	1.8	Nginx web site (https://nginx.org/download/)
Neo4j	2.3.1 Community	Neo4j web site (http://neo4j.com/download/)
Elasticsearch	2.3.5	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)
elasticsearch-dump	2.4.2	Install globally as a Node.js module (https://www.npmjs.com/package/elasticsearch-dump)
Logstash	2.3.4	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)
unzip	-	Install with yum install unzip 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)
poppler-utils	n/a	Install with yum install poppler-utils on CentOS/RHEL (https://apps.fedoraproject.org/packages/poppler-utils)



- As per *elasticsearch-dump* version 2.4.0, we recommended installing *elasticsearch-dump* as a **global Node.js module** (<https://www.npmjs.com/package/elasticsearch-dump#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 the dependencies

Install all required dependencies and third-party products before installing the platform.

To perform several tasks in the procedure, you may need root-level access rights.

To obtain administration rights, run the following command(s):

```
$ sudo su -
```

Add the repositories

- Add the following repositories to the system as root:

```
$ 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  
  
$ wget https://centos7.iuscommunity.org/ius-release.rpm && rpm -ivh ius-release.rpm  
  
$ 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>):

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


IUS repository

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

```
$ 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 8u74 or later:

```
$ wget -q --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/{JDK_release_number}/{auth_token}/jdk-
{JDK_release_number}-linux-x64.rpm"
```

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

```
# The JDK RPM install file name may change,
# this is just an example:

$ yum install jdk-8u131-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/scottvroenthal/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:

```
$ 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:

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

- Verify that Nginx is correctly installed:

```
$ type nginx
```

PostgreSQL

- Download **PostgreSQL 9.5** (<https://yum.postgresql.org/repopackages.php>):

```
$ 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** (<https://yum.postgresql.org/repopackages.php>):

```
$ 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:

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

- Start the PostgreSQL service:

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

- Check if PostgreSQL is running:

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

Configure PostgreSQL (1 of 2)

**Warning:**

You should configure PostgreSQL in 2 steps:

- Create the main data storage *before* installing the platform.
- Add the database to the platform settings *after* installing the platform.

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;
```

Set the database authentication method

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

```
$ 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	same-net	md5

Restart the database service

- Restart the PostgreSQL service:

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

or:

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

- Verify that user and database are successfully created:

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

Redis

- Install **Redis 3.2.3-1.el7** (<http://redis.io/download>):

```
$ 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:

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

- **Paste the following lines**

(<https://www.elastic.co/guide/en/elasticsearch/reference/2.3/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):

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

Autostart on boot

- **Enable Elasticsearch and set it up to automatically start on system boot:**

```
$ systemctl daemon-reload
$ systemctl enable elasticsearch
```

- **Start the Elasticsearch service:**

```
$ systemctl start elasticsearch
```

- **Verify that the Elasticsearch service is up and running:**

```
$ systemctl status elasticsearch
```

Install the Elasticsearch plugins

- **Install delete-by-query by following the procedure described on the **Elasticsearch 2.3 documentation****

(<https://www.elastic.co/guide/en/elasticsearch/plugins/2.3/plugins-delete-by-query.html>).

- Install elasticsearch 2.4.2:

```
$ npm install elasticsearch@2.4.2 -g
```

Logstash

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

```
$ 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>):

```
$ yum install logstash
```

Kibana

- Set up the **Kibana** (<https://www.elastic.co/guide/en/kibana/4.5/index.html>) repository by creating the *kibana.repo* repository file:

```
$ 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:

```
$ 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:

```
$ 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/testing
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:

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

poppler-utils

The platform requires **Poppler** (<https://poppler.freedesktop.org/>) to **process PDF** ([https://en.wikipedia.org/wiki/poppler_\(software\)#poppler-utils](https://en.wikipedia.org/wiki/poppler_(software)#poppler-utils)) file uploads. To install **poppler-utils** (<https://apps.fedoraproject.org/packages/poppler-utils>), do the following:

```
$ yum install poppler-utils
```


Install via an RPM package

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

To perform several tasks in the procedure, you may need root-level access rights. To obtain administration rights, run the following command(s):

```
$ sudo su -
```



Warning:

You should configure PostgreSQL in 2 steps:

- Create the main data storage *before* installing the platform.
- Add the database to the platform settings *after* installing 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	<code>\$ unzip eclecticiq-platform-x.x.x.zip</code>
Install the platform	<code>\$ yum install -y eclecticiq-platform</code>



- 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-1.14.2.zip* or *eclecticiq-platform-1.14.2.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:"

```
$ 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:

```
$ 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
$ 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):

```
$ 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 1.14.1, you need to upgrade step by step by installing all intermediate releases until 1.14.1.

Example:

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

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

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

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

# Upgrade from 1.13.0 to 1.14.0
$ yum install -y eclecticiciq-platform-1.14.0

# Upgrade from 1.14.0 to 1.14.1
$ yum install -y eclecticiciq-platform-1.14.1
```

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
```

<code>\${INSTALLED_VERSION}</code>	The currently installed version on your system. The upgrade fails because this version is too old.
<code>\${PREV_VERSION}</code>	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.
<code>exit 99</code>	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):

```
$ 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):

```
$ 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):

```
$ 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
$ setenforce 0

# Set SELinux to enforcing mode
$ 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

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	Description
/opt/eclecticiq/etc/eclecticiq/platform_settings.py	Contains core platform settings like security keys, external component URLs pointing to external components Celery-

File name and location	Description
/opt/eclecticiq/etc/eclecticiq/opentaxii.yml	Contains OpenTAXII (https://opentaxii.org/) 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 that are comma-separated. The no-proxy list needs to include 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 logs for the API, intel ingestion, and tasks.
/opt/eclecticiq/etc/logstash/conf.d/platform-ui.conf	Defines the locations of the log files storing logs for the web server; if it is not working normally, it is unavailable.
/opt/eclecticiq/etc/logstash/conf.d/input.conf	Elasticsearch Curator input configuration file. It is used to manage Elasticsearch indices. (https://www.elastic.co/guide/en/elasticsearch-curator/latest/)
/opt/eclecticiq/etc/logstash/conf.d/output.conf	Defines the Elasticsearch cluster and the data to be sent to Elasticsearch.
/opt/eclecticiq/etc/logstash/conf.d/filters.conf	Defines filters as needed to select specific indices.
/opt/eclecticiq/etc/logstash/conf.d/neo4j-batching.conf	Configuration file of the neo4j-batching process.
/opt/eclecticiq/etc/logstash/conf.d/opentaxii.conf	Defines the locations of the log files storing logs for the web server OpenTAXII relies on.
/opt/eclecticiq/etc/nginx/conf.d/platform.conf	Defines the platform configuration the Nginx web server uses, key, ports, endpoints to make available service documentation, and so on.
/opt/eclecticiq/etc/sysconfig/logstash	INI configuration file defining the Logstash heap size (GB).
/opt/eclecticiq/etc/supervisord.d/platform-api.ini	Supervisor INI configuration file to start the platform API.
/opt/eclecticiq/etc/supervisord.d/graph-ingestion.ini	Supervisor INI configuration file to start the graph ingestion.
/opt/eclecticiq/etc/supervisord.d/intel-ingestion.ini	Supervisor INI configuration file to start the intel ingestion.
/opt/eclecticiq/etc/supervisord.d/search-ingestion.ini	Supervisor INI configuration file to start the Elasticsearch ingestion.

File name and location	Description
/opt/eclecticiq/etc/supervisord.d/neo4j-batching.ini	Initializes the <code>neo4j-batching</code> process. By default, the process starts automatically.
/opt/eclecticiq/etc/supervisord.d/opentaxii.ini	Supervisor INI configuration file to start OpenTaxii incoming and outgoing feeds.
/opt/eclecticiq/etc/supervisord.d/task-workers.ini	Supervisor INI configuration file to start the Celery integrations, enrichers, utilities, and workers.
/opt/eclecticiq/etc-extras/kibana.yml	Kibana configuration file. Check it and edit or upgrade to a newer version.
/opt/eclecticiq/etc-extras/redis.conf	Defines the configuration for the Redis message broker snapshot autosave time intervals, and so on.
/opt/eclecticiq/etc-extras/elasticsearch/elasticsearch.yml	Elasticsearch configuration file. Check it and edit it to increase enough memory to the heap size (https://www.elastic.co/guide/en/elasticsearch/reference/current/settings.html) based on your system configuration and the system's recommended value is 2 GB : <code>ES_HEAP_SIZE=2GB</code> .
/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 and other options.
/opt/eclecticiq/etc-extras/neo4j/neo4j-server.properties	Defines the configuration options of the Neo4j server 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 and 10240.
/opt/eclecticiq/etc-extras/nginx/nginx.conf	Defines the configuration options for the Nginx web server.
/opt/eclecticiq/etc-extras/supervisord/kibana.ini	Initializes the <code>kibana</code> process. By default, the process starts automatically.
/opt/eclecticiq/etc-extras/supervisord/neo4j-console.ini	Initializes the <code>neo4j-console</code> process, that is used to start automatically.
/etc/supervisord.conf	Contains the Supervisor configuration (http://supervisord.org/configuration.html) for <code>supervisord</code> and <code>supervisorctl</code> . This file should be edited to enable the <code>supervisord</code> server to reply to <code>supervisorctl</code> commands.
/var/lib/pgsql/9.5/data/postgresql.conf	PostgreSQL configuration file (https://www.postgresql.org/docs/9.5/static/config.html).
/opt/eclecticiq/etc-extras/postfix/main.cf	Postfix configuration file (http://www.postfix.org/CONFIGURATION.html) for configuring email addresses through the GUI, 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
/var/log/eclecticiq/graph-ingestion.log
/var/log/eclecticiq/intel-ingestion.log
/var/log/eclecticiq/kibana.log
/var/log/eclecticiq/neo4j-batching.log
/var/log/eclecticiq/neo4j.log
/var/log/eclecticiq/opentaxii.log
/var/log/eclecticiq/platform-api.log
/var/log/eclecticiq/search-ingestion.log
/var/log/eclecticiq/task-beat.log
/var/log/eclecticiq/task-worker-enrichers.log
/var/log/eclecticiq/task-worker-integrations.log
/var/log/eclecticiq/task-worker-providers.log
/var/log/eclecticiq/task-worker-reindexing.log
/var/log/eclecticiq/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
/var/log/eclecticiq/platform-api.log	Platform log file. It logs core platform information.
/var/log/eclecticiq/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
/var/log/eclecticiq/intel-ingestion.log	Intel ingestion log file. It logs information on ingestion events, as well as ingested batches and blobs.
/var/log/eclecticiq/search-ingestion.log	Search indexing log file. It logs information on Elasticsearch data indexing.
/var/log/eclecticiq/kibana.log	It logs information on Kibana dashboard like connection and availability.
/var/log/eclecticiq/neo4j.log	Neo4j graph database log file. It logs information on Neo4j server and database operation.
/var/log/eclecticiq/opentaxii.log	It logs OpenTAXII server log information.
/var/log/eclecticiq/task-beat.log	It logs heartbeat timestamp information. The heartbeat interval is 30 seconds.
/var/log/eclecticiq/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.
/var/log/eclecticiq/task-worker-enrichers.log	It lists enricher tasks, intel providers, feed transport types, and platform utility tasks running in the background.
/var/log/eclecticiq/task-worker-integrations.log	It lists integrations like enricher tasks, intel providers, and platform utility tasks running in the background.
/var/log/eclecticiq/task-worker-providers.log	It lists integrations like enricher tasks, intel providers, and platform utility tasks running in the background.
/var/log/eclecticiq/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.

File name and location	Description
/var/log/--	This is the root directory where all log files generated by the platform and its (third-party) components are stored.

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

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

When performing system tasks such as installation, upgrade, or maintenance, you may need to access files and directories with a specific user access profile:

```
# run this command to login as root with current user/pw
$ sudo su -

# run this command to login as root with eclecticiq user/pw
$ su - eclecticiq

# run this command to login as root with elasticsearch user/pw
$ su - elasticsearch

# run this command to login as root with logstash user/pw
$ su - logstash

# run this command to login as root with neo4j user/pw
$ su - neo4j

# run this command to login as root with nginx user/pw
$ su - nginx

# run this command to login as root with postgres user/pw
$ su - postgres

# run this command to login as root with redis user/pw
$ su - 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

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 such as web server, graph, indexing, and search:

- Configure the database (step 2 of 2) (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.

To perform several tasks in the procedure, you may need root-level access rights. To obtain administration rights, run the following command(s):

```
$ sudo su -
```

Configure PostgreSQL (2 of 2)



Warning:

You should configure PostgreSQL in 2 steps:

- Create the main data storage *before* installing the platform.
- Add the database to the platform settings *after* installing the platform.

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:

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

# Example:
SQLALCHEMY_DATABASE_URI = "postgresql://test:test@localhost:5432/platform"
```

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

Restart the database service

- Restart the PostgreSQL service:

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

or:

```
$ 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 perform several tasks in the procedure, you may need root-level access rights.

To obtain administration rights, run the following command(s):

```
$ sudo su -
```

To configure Nginx, do the following:

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

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

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

```
$ 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/nginx_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/nginx_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:

```
$ 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:

```
$ 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:

```
$ service nginx reload
```

- Start the Nginx web server:

```
$ systemctl start nginx
```

or:

```
$ service nginx start
```

Configure Redis

To perform several tasks in the procedure, you may need root-level access rights.

To obtain administration rights, run the following command(s):

```
$ sudo su -
```

To configure Redis, do the following:

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

```
$ cp /opt/eclecticqiq/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:

```
$ 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:

```
$ 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):

```
$ service redis start
```

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

```
$ 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 perform several tasks in the procedure, you may need root-level access rights.

To obtain administration rights, run the following command(s):

```
$ sudo su -
```

To configure Elasticsearch, do the following:

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

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

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

```
$ 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:

```
$ 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:

```
$ 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
```

Example *elasticsearch.yml* file for reference:

```
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
```

Set heap size and custom tmp dir

You should allocate enough memory to the **heap size**

(<https://www.elastic.co/guide/en/elasticsearch/guide/current/heap-sizing.html>), based on your system configuration and the size of the Elasticsearch index.

The minimum recommended value is 2 GB: `ES_HEAP_SIZE=2g`.

It also a good idea to **change the default temp directory**

(<https://github.com/elastic/elasticsearch/issues/18406>) location and set a custom one.

You can set both options in the same configuration file:

- Go to `/etc/sysconfig` and open the `elasticsearch` configuration file:

```
$ nano /etc/sysconfig/elasticsearch
```

Example *elasticsearch* file for reference:

```
ES_HOME=/usr/share/elasticsearch
ES_HEAP_SIZE=2g
ES_JAVA_OPTS="-Djna.tmpdir=/<path_to>/<elasticsearch_tmp_dir>"
MAX_OPEN_FILES=65535
MAX_MAP_COUNT=262144
LOG_DIR=/var/log/elasticsearch
DATA_DIR=/media/elasticsearch
WORK_DIR=/tmp/elasticsearch
CONF_DIR=/etc/elasticsearch
ES_USER=elasticsearch
```

- Make sure you set the Elasticsearch heap size to at least 2 GB or more, if your system allows it:

```
ES_HEAP_SIZE=2g
```

- Set a custom temp directory by editing or adding the `ES_JAVA_OPTS` property:

```
ES_JAVA_OPTS="-Djna.tmpdir=/<path_to>/<elasticsearch_tmp_dir>"
```

- Verify that the `elasticsearch` user can read and write to the temp directory.

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):

```
$ 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:

```
$ 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:

```
$ 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.#"}]
```

**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.

Configure OpenTAXII

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

You can configure the platform to import users and groups from an LDAP directory service. It is preferable not to mix LDAP users and local users in the platform. If you create local users in the platform, and then import LDAP users, LDAP users override the local ones. When such an override occurs, the platform recognizes LDAP users as internal, that is, local, and local platform users as external.

A standard way to structure LDAP groups can include the following steps:

- Create an LDAP group to hold groups, an LDAP group to hold users, and an LDAP group to hold roles.
 - Apply some basic standard naming rules for these LDAP groups. For example, use a prefix such as `EclecticIQ` or `EIQ`.
Example:
 - `EclecticIQGroups`
 - `EclecticIQUsers`
 - `EclecticIQRoles`
- Create LDAP users as necessary.
- Assign these users to the `EclecticIQUsers` LDAP group.
- Create LDAP groups as necessary.
- The group names you define in the LDAP structure should match exactly any existing group names in the platform.
- Assign these groups to the `EclecticIQGroups` LDAP group.
- Create LDAP roles as necessary.
- The role names you define in the LDAP structure should match exactly the existing role names in the platform.
- Assign these roles to the `EclecticIQRoles` LDAP group.
- To define user group membership and user access policies, add the users to the child groups and the child roles of the `EclecticIQGroups` and `EclecticIQRoles` parent groups.

To configure LDAP authentication, do the following:

- Append the following attributes and set them to the appropriate values for your environment 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).
To enable LDAP authentication, set it to `True`.

- **LDAP_URI:** URI pointing to the designated LDAP instance. It supports `ldap://`, as well as `ldaps://` protocols.

Example:

```
LDAP_URI = "ldap://ldap.example.com"
```

- **LDAP_IGNORE_TLS_ERRORS:** Boolean switch to enable/disable TLS error checking such as invalid certificates, chain validation issues, and so on.
Default value: `True` (TLS errors are ignored).
- **LDAP_BIND_DN:** specify the valid credentials to bind to the LDAP connection, search for users, read groups and roles.
- **LDAP_BIND_PASSWORD:** specify the password associated to the binding credentials.

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 accounts.
The `{username}` placeholder value is replaced with an 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 replaced with an assigned user group 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 roles.
The `{username}` placeholder value is replaced with an assigned role name.
Example:

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

- **LDAP_USER_FIRSTNAME_ATTR:** define the naming attribute used to extract a user's first/given name.
(<http://ldapwiki.com/wiki/best%20practices%20for%20ldap%20naming%20attributes>)

- `LDAP_USER_LASTNAME_ATTR`: define the naming attribute used to extract a user's surname/family name.
- `LDAP_USER_EMAIL_ATTR`: define the naming attribute used to extract a user's email address.
- `LDAP_ROLE_NAME_ATTR`: define the naming attribute used to extract a user's role details.
- `LDAP_GROUP_NAME_ATTR`: define the naming attribute used to extract a user's group details.
- `LDAP_USER_IS_ADMIN_ATTR`: specify if the user should be granted admin rights.
To flag a user as an admin, assign the attribute the following value: `"isEclecticIQAdmin"`.
Example:

```
LDAP_USER_IS_ADMIN_ATTR = "isEclecticIQAdmin"
```

Example LDAP configuration:

```
# LDAP settings

LDAP_AUTH_ENABLED = True
LDAP_URI = "ldaps://10.0.2.212"
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"
LDAP_USER_IS_ADMIN_ATTR = "isEclecticIQAdmin"
```



LDAP referrals (<http://www.openldap.org/doc/admin24/referrals.html>) are not supported.

Configure LDAP to work with AD

You can set up LDAP authentication to work with AD (Active Directory).

EclecticIQ Platform provides a generic AD implementation. You can use it as a template to fine-tune attributes and parameters to suit the specific AD setup in your environment.

This table sums up the main differences between a vanilla LDAP configuration, and and LDAP setup that enables interoperability with AD:

LDAP	AD
-	memberOf:1.2.840.113556.1.4.1941
objectClass=posixGroup	objectClass=group
memberUid={username}	member={user_dn}
cn	cn, sAMAccountName, givenName

- **memberOf:1.2.840.113556.1.4.1941**: this string enables recursive filtering and match search. The magic number is an **OID** (<http://www.oid-info.com/cgi-bin/display?oid=1.2.840.113556.1.4.1941&action=display>) that identifies the **LDAP_MATCHING_RULE_IN_CHAIN** ([https://msdn.microsoft.com/en-us/library/windows/desktop/aa746475\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa746475(v=vs.85).aspx)) matching rule.
You need to include this string if you want to enable recursive pattern search inside a hierarchical data structure.
- **objectClass=group**: the group name that identifies AD groups, as opposed to **objectClass=posixGroup**, which represents Unix groups.
- **member={user_dn}: {user_dn}** takes the returned user object value. This is the full user **DN** (<http://ldapwiki.com/wiki/distinguished%20names>) that is filled after the first user-search operation.
- **cn, sAMAccountName, givenName**: naming attributes that can vary, depending on the specific AD setup in your environment.

Example

To perform several tasks in the procedure, you may need root-level access rights.

To obtain administration rights, run the following command(s):

```
$ sudo su -
```

- Append the following parameters to `/opt/eclecticiq/etc/eclecticiq/platform_settings.py`.
- Restart the `platform-api` service:


```
$ supervisorctl restart platform-api
```

LDAP/AD configuration enabling users to sign in with their designated user name (example: *k_soze*)

LDAP_USERS_FILTER = "sAMAccountName={username}" sets signing in with the user's user name.

```
LDAP_AUTH_ENABLED = True
LDAP_URI = 'ldap://10.0.12.154'

LDAP_BIND_DN = "admin@example.com"
LDAP_BIND_PASSWORD = "=BKHpUW3f="

LDAP_USERS_FILTER = (
    "cn=Users,dc=eiq,dc=local",
    "sAMAccountName={username}")

LDAP_GROUPS_FILTER = (
    "cn=Users,dc=eiq,dc=local",
    "(&
(memberOf:1.2.840.113556.1.4.1941:=cn=EclecticIQGroups,cn=Users,dc=eiq,dc=local) "
    "(objectClass=group) "
    "(member={user_dn}))")

LDAP_ROLES_FILTER = (
    "cn=Users,dc=eiq,dc=local",
    "(&
(memberOf:1.2.840.113556.1.4.1941:=cn=EclecticIQRoles,cn=Users,dc=eiq,dc=local) "
    "(objectClass=group) "
    "(member={user_dn}))")

LDAP_USER_FIRSTNAME_ATTR = 'givenName'
LDAP_USER_LASTNAME_ATTR = 'sn'
LDAP_USER_EMAIL_ATTR = 'mail'
LDAP_ROLE_NAME_ATTR = 'sAMAccountName'
LDAP_GROUP_NAME_ATTR = 'sAMAccountName'
```

LDAP/AD configuration enabling users to sign in with their full name (example: *Keyser Söze*)

LDAP_USERS_FILTER = "cn={username}" sets signing in with the user's full name.

```
LDAP_AUTH_ENABLED = True
LDAP_URI = 'ldap://10.0.12.154'

LDAP_BIND_DN = "admin@example.com"
LDAP_BIND_PASSWORD = "=BKHpUW3f="

LDAP_USERS_FILTER = (
    "cn=Users,dc=eiq,dc=local",
    "cn={username}")

LDAP_GROUPS_FILTER = (
    "cn=Users,dc=eiq,dc=local",
    " (&
(memberOf:1.2.840.113556.1.4.1941:=cn=EclecticIQGroups,cn=Users,dc=eiq,dc=local) "
    "(objectClass=group) "
    "(member={user_dn})) ")

LDAP_ROLES_FILTER = (
    "cn=Users,dc=eiq,dc=local",
    " (&
(memberOf:1.2.840.113556.1.4.1941:=cn=EclecticIQRoles,cn=Users,dc=eiq,dc=local) "
    "(objectClass=group) "
    "(member={user_dn})) ")

LDAP_USER_FIRSTNAME_ATTR = 'givenName'
LDAP_USER_LASTNAME_ATTR = 'sn'
LDAP_USER_EMAIL_ATTR = 'mail'
LDAP_ROLE_NAME_ATTR = 'sAMAccountName'
LDAP_GROUP_NAME_ATTR = 'sAMAccountName'
```

Bootstrap the platform

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):

```
$ yum info supervisor
```

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

```
$ 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:

```
$ service supervisord start
```

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

```
$ 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 `$ supervisorctl reload` to update it and to reload the up-to-date configuration into the platform. Alternatively, run first `$ supervisorctl reread` to reload the changed configuration, and then `$ supervisorctl update` to restart the managed applications whose configurations have changed.

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.

```
$ INTELWORKS_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 do the following::

- 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 *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):

```
$ service neo4j start
```

or:

```
$ supervisorctl start neo4j
```

Create the graph schema

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

```
$ supervisorctl stop graph-ingestion
$ 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 platform 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):

```
$ service nginx status
```

Supervisor

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

```
$ service supervisord status
```

PostgreSQL

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

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

or:

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

Check search indexing and graph

Elasticsearch

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

```
$ 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):

```
$ service neo4j start
```

or:

```
$ supervisorctl start neo4j
```

Reload the Supervisor configuration

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

```
$ 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 `$ supervisorctl reload` to update it and to reload the up-to-date configuration into the platform. Alternatively, run first `$ supervisorctl reread` to reload the changed configuration, and then `$ supervisorctl update` to restart the managed applications whose configurations have changed.

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

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.



To perform several tasks in the procedure, you may need root-level access rights. To obtain administration rights, run the following command(s):

```
$ sudo su -
```

The upgrade procedure consists of the following steps:

Before the upgrade

- Exit the platform
- Back up your data
- Shut down the platform
- 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.

Shut down the platform

To gracefully shut down EclecticIQ Platform, do the following:

- Stop all running processes.
- Shut down the machine the platform is hosted on.

Generic shutdown

You can stop the platform without following any specific procedure. However, if you want to make sure the platform core services and processes gracefully shut down, manually execute the following commands:

```
$ supervisorctl stop all
```

```
$ systemctl stop postgresql-9.5 redis logstash elasticsearch
```

You can then proceed to shut down the virtual or physical machine hosting the platform.

Shutdown before a platform upgrade

If you are shutting down the platform before performing an upgrade, stop platform components in the order described below to make sure no Celery tasks are left over in the queue. This prevents hanging tasks in the queue from interfering with the upgrade procedure.

- Stop platform-api:

```
$ supervisorctl stop platform-api
```

- Stop the Celery beat:

```
$ supervisorctl stop task:beat
```

- Check Celery queues — they should be empty:

```
$ redis-cli llen enrichers  
  
$ redis-cli llen integrations  
  
$ redis-cli llen priority_enrichers  
  
$ redis-cli llen priority_providers  
  
$ redis-cli llen priority_utilities  
  
$ redis-cli llen providers  
  
$ redis-cli llen reindexing  
  
$ redis-cli llen utilities
```

Alternatively, check Redis keys:

```
$ redis-cli keys *
```

If the response *does not include* any keys related to any Celery queues, they are empty and you can continue.

- Stop the remaining Celery workers:

```
$ supervisorctl stop task:*
```

- Stop Supervisor-managed workers:

- kibana
- intel-ingestion
- opentaxii
- search-ingestion
- graph-ingestion
- neo4j

```
$ supervisorctl stop all
```

Alternatively, you can list them explicitly:

```
$ supervisorctl stop kibana intel-ingestion opentaxii search-ingestion graph-ingestion  
neo4j
```

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 in the procedure, you may need root-level access rights.
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):

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

or:

```
$ 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>):

```
$ 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

Install the new package

You can now proceed with 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 1.14.1, you need to upgrade step by step by installing all intermediate releases until 1.14.1.

Example:


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

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

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

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

# Upgrade from 1.13.0 to 1.14.0
$ yum install -y eclecticiciq-platform-1.14.0

# Upgrade from 1.14.0 to 1.14.1
$ yum install -y eclecticiciq-platform-1.14.1
```

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
```

<code>\${INSTALLED_VERSION}</code>	The currently installed version on your system. The upgrade fails because this version is too old.
<code>\${PREV_VERSION}</code>	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.
<code>exit 99</code>	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.

<code>platform_settings.py</code>	The main configuration file for the platform
<code>opentaxii.yml</code>	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 Community: 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/>).

Install poppler-utils

The platform requires **Poppler** (<https://poppler.freedesktop.org/>) to **process PDF** ([https://en.wikipedia.org/wiki/poppler_\(software\)#poppler-utils](https://en.wikipedia.org/wiki/poppler_(software)#poppler-utils)) file uploads. To install **poppler-utils** (<https://apps.fedoraproject.org/packages/poppler-utils>), do the following:

```
$ yum install poppler-utils
```

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.

```
$ INTELWORKS_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:

```
$ supervisorctl stop graph-ingestion
$ 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/ &&
INTELWORKS_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):

```
$ service nginx status
```

Supervisor

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

```
$ service supervisord status
```

PostgreSQL

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

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

or:

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

Check search indexing and graph

Elasticsearch

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

```
$ 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):

```
$ service neo4j start
```

or:

```
$ supervisorctl start neo4j
```

Reload the Supervisor configuration

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

```
$ 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

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 specific 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, observables, 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 initialization. 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):

```
$ service neo4j stop
```

- Make sure Neo4j has stopped.
Run the following command(s):

```
$ 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** (<https://neo4j.com/docs/operations-manual/current/backup/#backup-restoring>) **Neo4j database backup**

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.

How to install the platform via an RPM package

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 EclecticlQ 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:

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

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

```
$ 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:

```
$ yum check-update
```

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

```
$ yum update
```

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

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

- Make sure your RHEL or CentOS server includes **supervisord** (<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):

```
$ 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:

```
$ systemctl status supervisord
```

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

```
$ systemctl enable supervisord
```

- After enabling *supervisord*, start it:

```
$ systemctl start supervisord
```

Add the repositories

- Add the following repositories to the system as root:

```
$ 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
```

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

```
$ 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>):

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

IUS repository

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

```
$ 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 8u74 or later:

```
$ wget -q --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/{JDK_release_number}/{auth_token}/jdk-
{JDK_release_number}-linux-x64.rpm"
```

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

```
# The JDK RPM install file name may change,
# this is just an example:
```

```
$ yum install jdk-8u131-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/scottvroenthal/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:

```
$ 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:

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

- Verify that Nginx is correctly installed:

```
$ type nginx
```

PostgreSQL

- Download **PostgreSQL 9.5** (<https://yum.postgresql.org/repopackages.php>):

```
$ 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** (<https://yum.postgresql.org/repopackages.php>):

```
$ 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:

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

- Start the PostgreSQL service:

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

- Check if PostgreSQL is running:

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

Configure PostgreSQL (1 of 2)

**Warning:**

You should configure PostgreSQL in 2 steps:

- Create the main data storage *before* installing the platform.
- Add the database to the platform settings *after* installing the platform.

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;
```

Set the database authentication method

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

```
$ 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	same	md5

Restart the database service

- Restart the PostgreSQL service:

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

or:

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

- Verify that user and database are successfully created:

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

Redis

- Install **Redis 3.2.3-1.el7** (<http://redis.io/download>):

```
$ 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:

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

- **Paste the following lines**

(<https://www.elastic.co/guide/en/elasticsearch/reference/2.3/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):

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

Autostart on boot

- **Enable Elasticsearch and set it up to automatically start on system boot:**

```
$ systemctl daemon-reload
$ systemctl enable elasticsearch
```

- **Start the Elasticsearch service:**

```
$ systemctl start elasticsearch
```

- **Verify that the Elasticsearch service is up and running:**

```
$ systemctl status elasticsearch
```

Install the Elasticsearch plugins

- **Install delete-by-query by following the procedure described on the **Elasticsearch 2.3 documentation****

(<https://www.elastic.co/guide/en/elasticsearch/plugins/2.3/plugins-delete-by-query.html>).

- Install elasticsearch 2.4.2:

```
$ npm install elasticsearch@2.4.2 -g
```

Logstash

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

```
$ 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>):

```
$ yum install logstash
```

Kibana

- Set up the **Kibana** (<https://www.elastic.co/guide/en/kibana/4.5/index.html>) repository by creating the *kibana.repo* repository file:

```
$ 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:

```
$ 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:

```
$ 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/testing
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:

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

Install poppler-utils

The platform requires **Poppler** (<https://poppler.freedesktop.org/>) to **process PDF** ([https://en.wikipedia.org/wiki/poppler_\(software\)#poppler-utils](https://en.wikipedia.org/wiki/poppler_(software)#poppler-utils)) file uploads. To install **poppler-utils** (<https://apps.fedoraproject.org/packages/poppler-utils>), do the following:

```
$ yum install poppler-utils
```

Set up the Eclectiq repository

- Set up the Eclectiq repository by creating the *eclectiq-platform.repo* repository file:

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

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

```
[eclectiq-platform]
name=eclectiq-platform
baseurl=https://downloads.eclectiq.com/platform
gpgcheck=1
repo_gpgcheck=0
enabled=1
username=<username>
password=<password>
gpgkey=https://downloads.eclectiq.com/platform/repokey/repokey.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-eclectiq.repo*.
In case it does not happen automatically, you can safely remove */etc/yum.repos.d/private-eclectiq.repo* manually:"

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

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

```
$ 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 *eclectiq-platform-1.14.2.zip* or *eclectiq-platform-1.14.2.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.)

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

# For example, to install release 1.14.2:
$ yum -y install eclecticiq-platform-1.14.2
```

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

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

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

```
$ 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
```

Configure PostgreSQL (2 of 2)



Warning:

You should configure PostgreSQL in 2 steps:

- Create the main data storage *before* installing the platform.
- Add the database to the platform settings *after* installing the platform.

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:


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

# Example:
SQLALCHEMY_DATABASE_URI = "postgresql://test:test@localhost:5432/platform"
```

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

Restart the database service

- Restart the PostgreSQL service:

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

or:

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

- Verify that user and database are successfully created:

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

Configure Nginx

To perform several tasks in the procedure, you may need root-level access rights. To obtain administration rights, run the following command(s):

```
$ sudo su -
```

To configure Nginx, do the following:

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

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

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

```
$ 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/nginx_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/nginx_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:

```
$ 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:

```
$ 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:

```
$ service nginx reload
```

Start Nginx

- Start the Nginx web server:

```
$ systemctl start nginx
```

or:

```
$ service nginx start
```

Configure Redis

To perform several tasks in the procedure, you may need root-level access rights. To obtain administration rights, run the following command(s):

```
$ sudo su -
```

To configure Redis, do the following:

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

```
$ 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:

```
$ 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:

```
$ 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):

```
$ service redis start
```

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

```
$ 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 perform several tasks in the procedure, you may need root-level access rights. To obtain administration rights, run the following command(s):

```
$ sudo su -
```

To configure Elasticsearch, do the following:

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

```
$ cp /opt/eclectic iq/etc-extras/elasticsearch/elasticsearch.yml
/etc/elasticsearch/elasticsearch.yml
```

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

```
$ 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:

```
$ 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:

```
$ 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
```

Example *elasticsearch.yml* file for reference:

```
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
```

Set heap size and custom tmp dir

You should allocate enough memory to the **heap size**

(<https://www.elastic.co/guide/en/elasticsearch/guide/current/heap-sizing.html>), based on your system configuration and the size of the Elasticsearch index.

The minimum recommended value is 2 GB: `ES_HEAP_SIZE=2g`.

It also a good idea to **change the default temp directory**

(<https://github.com/elastic/elasticsearch/issues/18406>) location and set a custom one.

You can set both options in the same configuration file:

- Go to `/etc/sysconfig` and open the `elasticsearch` configuration file:

```
$ nano /etc/sysconfig/elasticsearch
```

Example *elasticsearch* file for reference:

```
ES_HOME=/usr/share/elasticsearch
ES_HEAP_SIZE=2g
ES_JAVA_OPTS="-Djna.tmpdir=/<path_to>/<elasticsearch_tmp_dir>"
MAX_OPEN_FILES=65535
MAX_MAP_COUNT=262144
LOG_DIR=/var/log/elasticsearch
DATA_DIR=/media/elasticsearch
WORK_DIR=/tmp/elasticsearch
CONF_DIR=/etc/elasticsearch
ES_USER=elasticsearch
```

- Make sure you set the Elasticsearch heap size to at least 2 GB or more, if your system allows it:

```
ES_HEAP_SIZE=2g
```

- Set a custom temp directory by editing or adding the `ES_JAVA_OPTS` property:

```
ES_JAVA_OPTS="-Djna.tmpdir=/<path_to>/<elasticsearch_tmp_dir>"
```

- Verify that the `elasticsearch` user can read and write to the temp directory.

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 => ["/var/log/eclecticiq/platform-api.log", "/var/log/nginx/eclecticiq-
platform-api-nginx-access.log", "/var/log/eclecticiq/intel-ingestion.log",
"/var/log/eclecticiq/graph-ingestion.log", "/var/log/eclecticiq/search-ingestion.log"]
    codec => "json"
    tags => "platform-api"
  }
  file {
    path => ["/var/log/eclecticiq/intel-ingestion.log", "/var/log/eclecticiq/graph-
ingestion.log", "/var/log/eclecticiq/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 => ["/var/log/eclecticiq/task-worker-*.log", "/var/log/eclecticiq/task-
beat.log", "/var/log/eclecticiq/task-queue-analysis-listener.log",
"/var/log/eclecticiq/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.

To perform several tasks in the procedure, you may need root-level access rights.

To obtain administration rights, run the following command(s):

```
$ sudo su -
```

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

```
$ 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 transformations.
- `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.html#encodes>) 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:

```
$ systemctl enable logstash
```

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

```
$ 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):

```
$ 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:

```
$ service supervisord restart
```

or:

```
$ systemctl restart supervisord
```

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

Alternatively, run first `$ supervisorctl reread` to reload the changed configuration, and then `$ supervisorctl update` to restart the managed applications whose configurations have changed.

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

```
$ supervisorctl reload
```

Start Kibana

- You can now start Kibana:

```
$ service kibana start
```

or:

```
$ systemctl start kibana
```

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

```
$ service kibana status
```

or:

```
$ 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:

```
$ 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:

```
$ 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:

```
$ service supervisord restart
```

or:

```
$ systemctl restart supervisord
```

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

```
$ 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
$ service elasticsearch stop

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

- Start Elasticsearch:

```
# Start Elasticsearch
$ 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 *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):

```
$ service neo4j start
```

or:

```
$ supervisorctl start neo4j
```

- Verify that Neo4j is running:

```
$ service neo4j status
```

Create the graph schema

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

```
$ supervisorctl stop graph-ingestion
$ 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 elasticsearchdump to load the platform dashboard
$ elasticsearchdump --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):

```
$ 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.



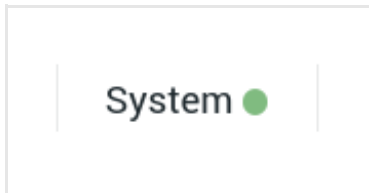
Warning: The browser may display an untrusted connection warning: add it as an exception, and then proceed to the platform.

How to check system health

System health gives you a clear basic overview of the overall platform health, as well as the operational status of its components.

Check system health via the GUI

The first and easiest way to check normal platform operation is by using the GUI system health tool on the status bar:



A green or red dot next to **System** on the status bar indicates the global system health status. Click **System** to drill down to a platform component-level. On the pop-up window click either **Processes** or **Services** to view the status of the normal platform processes or of the platform core services.

System Health ●



PROCESSES

SERVICES

Name	Processes	Status
newrelic_redis_agent	1	●
task	5	●
intel-ingestion	4	●
neo4j-batching	1	●
newrelic_postgresql_agent	1	●
newrelic_elasticsearch_agent	1	●
neo4j-console	1	●
opentaxii	1	●
kibana	1	●
graph-ingestion	1	●
platform-api	1	●
search-ingestion	1	●

System Health ●



PROCESSES

SERVICES

Name	Status
postgresql-9.4	● active
logstash	● active
elasticsearch	● active
redis	● active

Process	Description
graph-ingestion	Data funnel to the Neo4j graph database. Handles data updates for Neo4j
intel-ingestion	Intel ingestion through feeds and enrichers. Consumes incoming data and saves it PostgreSQL, Neo4j, and Elasticsearch. The platform executes one <code>intel-ingestion</code> per processor core. Running tasks are sequentially numbered starting from 0. For example, a platform instance running on a quad core machine normally executes 4 such processes, progressively numbered from <code>intel-ingestion:0</code> to <code>intel-ingestion:3</code>
kibana	Generates dashboard graphs
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-console	The platform uses Neo4j via this console. Neo4j is available through the platform as a Linux service
newrelic_elasticsearch_agent	Starts/Restarts Elasticsearch services, including data logging to monitor component health
newrelic_postgresql_agent	Starts/Restarts PostgreSQL services, including data logging
newrelic_redis_agent	Starts/Restarts Redis (message broker) services, including data logging
opentaxii	TAXII server responsible for STIX data transport

Process	Description
platform-api	The web application implementing the platform API and the API endpoints. The endpoints expose services that can be consumed by making API calls and by passing arguments
search-ingestion	Search indexer. Handles Elasticsearch data updates
task	Celery-managed tasks like enrichers, feed integrations, incoming feed data providers, and utilities

Service	Description
elasticsearch	Elasticsearch search and indexing database
postgresql-9.5	PostgreSQL (intel database)
redis	Redis (message broker)
logstash	Log and data aggregation, data pipeline and funneling
nginx	Web server

Check system health via the API

You can retrieve system health information also by making an API call to the `/status` API endpoint.



First, make an authentication call to receive the bearer token you need to pass with all subsequent API calls.

API endpoint

API endpoint	<code>/status</code>
Create method	GET
HTTP headers	"Content-Type: application/json", "Accept: application/json", "Authorization: Bearer <token>"

API request	GET + "Content-Type: application/json" + "Accept: application/json" + "Authorization: Bearer <token>" + <platform_host>/status
API response	{ "data" : { <status_reponse> } }

When you make a GET request and obtain a JSON object with entity data in the response, the entity object is wrapped in a data wrapper: { "data" : { ... } }.

Status request

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



Warning:

About cURL calls

- If you make HTTPs cURL calls to the API *and* you have a self-signed or an invalid certificate, include the `-k` or the `--insecure` parameter in the cURL call to skip the SSL connection CA certificate check.
- Always append a / trailing slash at the end of an API URL endpoint. The only exception is `/auth`, which does not take a trailing slash.
- In the cURL call, the `-d` data payload with the entity information always needs to be flat JSON, not hierarchical JSON.
If you want to pass a hierarchical JSON object, include the `--data-binary` parameter, followed by the path to the JSON file, for example `@/path/to/entity_file.json`.

Status response

```
{
  "data": {
    "health": "GREEN",
```

```
"process_states": [  
  
  {  
    "first_down_at": null,  
    "first_up_at": "2016-06-30T11:28:30+00:00",  
    "health": "GREEN",  
    "n_processes": 1,  
    "n_processes_down": 0,  
    "name": "newrelic_redis_agent"  
  },  
  
  {  
    "first_down_at": null,  
    "first_up_at": "2016-07-06T20:03:27+00:00",  
    "health": "GREEN",  
    "n_processes": 5,  
    "n_processes_down": 0,  
    "name": "task"  
  },  
  
  {  
    "first_down_at": null,  
    "first_up_at": "2016-07-06T20:04:36+00:00",  
    "health": "GREEN",  
    "n_processes": 4,  
    "n_processes_down": 0,  
    "name": "intel-ingestion"  
  },  
  
  {  
    "first_down_at": null,  
    "first_up_at": "2016-07-06T20:05:07+00:00",  
    "health": "GREEN",  
    "n_processes": 1,  
    "n_processes_down": 0,  
    "name": "neo4j-batching"  
  },  
  
  {  
    "first_down_at": null,  
    "first_up_at": "2016-06-30T11:28:35+00:00",  
    "health": "GREEN",  
    "n_processes": 1,  
    "n_processes_down": 0,  
    "name": "newrelic_postgresql_agent"  
  },  
  
  {  
    "first_down_at": null,  
    "first_up_at": "2016-06-30T11:28:35+00:00",  
    "health": "GREEN",  
    "n_processes": 1,  
    "n_processes_down": 0,  
    "name": "newrelic_elasticsearch_agent"  
  },  
]
```

```
{
  "first_down_at": null,
  "first_up_at": "2016-07-06T20:00:14+00:00",
  "health": "GREEN",
  "n_processes": 1,
  "n_processes_down": 0,
  "name": "neo4j-console"
},

{
  "first_down_at": null,
  "first_up_at": "2016-06-30T11:28:35+00:00",
  "health": "GREEN",
  "n_processes": 1,
  "n_processes_down": 0,
  "name": "opentaxii"
},

{
  "first_down_at": null,
  "first_up_at": "2016-07-01T06:54:15+00:00",
  "health": "GREEN",
  "n_processes": 1,
  "n_processes_down": 0,
  "name": "kibana"
},

{
  "first_down_at": null,
  "first_up_at": "2016-07-06T20:04:58+00:00",
  "health": "GREEN",
  "n_processes": 1,
  "n_processes_down": 0,
  "name": "graph-ingestion"
},

{
  "first_down_at": null,
  "first_up_at": "2016-07-06T20:03:21+00:00",
  "health": "GREEN",
  "n_processes": 1,
  "n_processes_down": 0,
  "name": "platform-api"
},

{
  "first_down_at": null,
  "first_up_at": "2016-07-06T20:05:04+00:00",
  "health": "GREEN",
  "n_processes": 1,
  "n_processes_down": 0,
  "name": "search-ingestion"
}
],
```

```
"service_states": [  
  {  
    "health": "GREEN",  
    "name": "postgresql-9.5",  
    "state": "active"  
  },  
  
  {  
    "health": "GREEN",  
    "name": "logstash",  
    "state": "active"  
  },  
  
  {  
    "health": "GREEN",  
    "name": "elasticsearch",  
    "state": "active"  
  },  
  
  {  
    "health": "GREEN",  
    "name": "redis",  
    "state": "active"  
  }  
]  
}
```


How to monitor the platform

As a system administrator, you can use tools like Celery and Supervisor to monitor platform tasks to check day-to-day operations and to investigate, in case an issue occurs.

Prerequisites

This article assumes that the following products are installed and configured in a production and/or in a test environment:

- EclecticIQ Platform
- Celery
- Redis
- Supervisord
- systemd

The EclecticIQ Platform runs on CentOS and RHEL; therefore, all command and data output examples apply to these Linux environments.

Scope

Monitor and inspect the status of key platform processes like incoming and outgoing feeds, enrichers, and tasks.

In the current context, monitoring covers on/off status only: the tasks and the commands here allow you to verify whether a task, a process, or a component is running or not. Metrics and other measurements are outside the scope of the topic.

Goal

Allow system administrators and devops engineers to execute quick checks to inspect platform operation, identify any issues and review errors, so that they can address them in a timely manner.

Audience

Tools

Celery	The task runner. It manages task execution and scheduling.
Redis	The message broker. It handles background task processing by managing message queues based on the pub-sub pattern (https://en.wikipedia.org/wiki/publish%e2%80%93subscribe_pattern).
Supervisor	The process controller to start and stop processes.
systemd	The initialization system to bootstrap processes and start services.

Core components

Component	Address	Port
nginx	<server_name> (http://nginx.org/en/docs/http/nginx_http_core_module.html#server_name)	80; 443
platform	localhost	8008
postgresql	localhost	5432
redis	localhost	6379
neo4j	localhost	7474
elasticsearch	localhost	9200
kibana	localhost	5601
logstash	localhost	6755

The PostgreSQL default database name is `platform-api`.

Monitoring

Platform monitoring covers two main areas:

Components	
platform-api	The web application implementing the platform API and the API endpoints. The endpoints expose services that can be consumed by making API calls and by passing arguments
graph-ingestion	Data funnel to the Neo4j graph database. Handles data updates for Neo4j
intel-ingestion	Intel ingestion through feeds and enrichers. Consumes incoming data and saves it PostgreSQL, Neo4j, and Elasticsearch. The platform executes one <code>intel-ingestion</code> per processor core. Running tasks are sequentially numbered starting from 0. For example, a platform instance running on a quad core machine normally executes 4 such processes, progressively numbered from <code>intel-ingestion:0</code> to <code>intel-ingestion:3</code>
opentaxii	TAXII server responsible for STIX data transport
postgresql-9.5	PostgreSQL (intel database)
redis	Redis (message broker)
neo4j-console	The platform uses Neo4j via this console. Neo4j is available through the platform as a Linux service
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
elasticsearch	Elasticsearch search and indexing database
task	Celery-managed tasks like enrichers, feed integrations, incoming feed data providers, and utilities

Processes	
Feeds	Incoming and outgoing feeds
Enrichers	Enricher tasks
Celery tasks	Other/Misc. Celery tasks

Monitor components with Supervisor

Tool: *Supervisor* is your first and most useful tool to inspect platform components to verify if they are operating normally.

Supervisor configuration files are stored here:

File	Location
platform-api.ini	/opt/eclecticiq/etc/supervisord.d/
graph-ingestion.ini	/opt/eclecticiq/etc/supervisord.d/
intel-ingestion.ini	/opt/eclecticiq/etc/supervisord.d/
search-ingestion.ini	/opt/eclecticiq/etc/supervisord.d/
neo4j-batching.ini	/opt/eclecticiq/etc/supervisord.d/
opentaxii.ini	/opt/eclecticiq/etc/supervisord.d/
task-workers.ini	/opt/eclecticiq/etc/supervisord.d/
kibana.ini	/opt/eclecticiq/etc-extras/supervisord/
neo4j-console.ini	/opt/eclecticiq/etc-extras/supervisord/

Use it to check the following components:

Component	Description	If it is not running...
graph-ingestion	Data funnel to the Neo4j graph database. Handles data updates for Neo4j	The graph database may go out of sync and miss data. The <code>queue:graph:inbound</code> queue increases in size
intel-ingestion	Intel ingestion through feeds and enrichers. Consumes incoming data and saves it PostgreSQL, Neo4j, and Elasticsearch. The platform executes one <code>intel-ingestion</code> per processor core. Running tasks are sequentially numbered starting from 0. For example, a platform instance running on a quad core machine normally executes 4 such processes, progressively numbered from <code>intel-ingestion:0</code> to <code>intel-ingestion:3</code>	Provider tasks keep running, but no data is displayed in the system. The <code>queue:ingestion:inbound</code> queue increases in size

Component	Description	If it is not running...
search-ingestion	Search indexer. Handles Elasticsearch data updates	Search indexing stops working correctly. Elasticsearch may go out of sync and miss data. The <code>queue:search:inbound</code> queue increases in size
kibana	Generates dashboard graphs	The dashboard does not load correctly, and the <code>/kibana/</code> API endpoint returns an HTTP 502 error
neo4j-console	The platform uses Neo4j via this console. Neo4j is available through the platform as a Linux service	Graph queries stop working, it is not possible to poll the graph database
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	Graph ingestion stops working, queries may return results that are not up to date
opentaxii	TAXII server responsible for STIX data transport	The TAXII transport type becomes unavailable
platform-api	The web application implementing the platform API and the API endpoints. The endpoints expose services that can be consumed by making API calls and by passing arguments	Platform services become unavailable and the API endpoints return an HTTP 502 error
task	Celery-managed tasks like enrichers, feed integrations, incoming feed data providers, and utilities	
task:beat	Task scheduler	Task execution order may be affected, and tasks may or may not start or stop, resulting in unexpected task execution behavior
task:enrichers	Enricher tasks	Enricher data may become only partially available or completely unavailable
task:integrations	Outgoing feed integrations	Outgoing feed transport types are affected, and they may stop working correctly or become unavailable

Component	Description	If it is not running...
<code>task:providers</code>	Incoming feed data provider tasks	Incoming feed transport types are affected, and they may stop working correctly or become unavailable
<code>task:utilities</code>	The little platform elves that keep things tidy while working in the background	The platform may behave unexpectedly. For example data updates and discovery may hang or stop working

`supervisorctl` is Supervisor's command line interface utility. The commands you can pass are called *actions*, and they can optionally take *arguments*:

```
$ supervisorctl <action> <argument>
```



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

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

```
$ yum info supervisor
```

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

```
$ yum install supervisor
```

This installs two components:

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

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/`.)

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

```
$ service supervisord status
```

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

```
$ service supervisord start
```

If you need to stop Supervisor, run the following command(s):

```
$ service supervisord stop
```

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
```

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

```
# Retrieve the status of a specific process
$ supervisorctl status <process_name>

# Retrieve the status of multiple specific processes
$ supervisorctl status <process_name> <process_name> <process_name> ...

# Retrieve the status of all processes
# whose name contains the specified search string
$ supervisorctl status | grep "<search_string>"
```

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

```
$ supervisorctl reload
```



Warning: When you modify or update the Supervisor configuration, you need to run `$ supervisorctl reload` to update it and to reload the up-to-date configuration into the platform. Alternatively, run first `$ supervisorctl reread` to reload the changed configuration, and then `$ supervisorctl update` to restart the managed applications whose configurations have changed.

Supervisor actions

These are examples of `supervisorctl` actions:

Run this...	...to do this
<code>\$ supervisorctl start all</code>	Start all the processes defined in the Supervisor configuration
<code>\$ supervisorctl start <process_name></code>	Start the specified process defined in the Supervisor configuration
<code>\$ supervisorctl start <process_name> <process_name></code>	Start the specified processes defined in the Supervisor configuration
<code>\$ supervisorctl stop all</code>	Stop all the processes defined in the Supervisor configuration file
<code>\$ supervisorctl stop <process_name></code>	Stop the specified process
<code>\$ supervisorctl stop <process_name> <process_name></code>	Stop the specified processes
<code>\$ supervisorctl status</code>	Retrieve the statuses of the processes managed by Supervisor. For further information, see the official ocumentation on the return state values (http://supervisord.org/subprocess.html#process-states)
<code>\$ supervisorctl status <process_name></code>	Retrieve the status of the specified process
<code>\$ supervisorctl status <process_name> <process_name></code>	Retrieve the status of the specified processes
<code>\$ supervisorctl reload</code>	Reload the Supervisor configuration and restart all tasks and processes. If you modify or update the Supervisor configuration file, you need run this command to reload the latest Supervisor configuration file

Run this...	...to do this
<code>\$ supervisorctl reload all</code>	Reload all the processes managed by Supervisor. This command works just like <code>\$ supervisorctl reload</code>
<code>\$ supervisorctl update</code>	Restart the applications whose configuration has changed. This command affects existing configurations that were modified. If new application configurations become available, they start automatically only after restarting Supervisor or after rebooting the system. Typically, you run <code>reread</code> and then <code>update</code> .
<code>\$ supervisorctl tail <log_file_name></code>	Retrieve the most recent lines of the specified log file. To follow the log file as it updates with new information and new lines, run <code>\$ supervisorctl tail -f <log_file_name></code>
<code>\$ supervisorctl status grep "<search_string>"</code>	Retrieve the status of all processes whose name contains the specified search string.
<code>\$ supervisorctl reread</code>	Update the changed configurations and reload them. It does not automatically (re)start any applications. Typically, you run <code>reread</code> and then <code>update</code> .

Monitor components with systemd

Tool: *systemd* helps you inspect platform components to verify if their services are running normally. Use it to check the following components:

Component	Description	If it is not running...
postgresql-9.5	PostgreSQL (intel database)	It is not possible to access platform data
redis	Redis (message broker)	Tasks and processes may hang and/or behave unexpectedly
elasticsearch	Elasticsearch search and indexing database	No data searching and indexing capabilities are available
logstash	Log and data aggregation, data pipeline and funneling	No data aggregation, deduplication, and normalization
nginx	Web server	The platform is down

`systemctl` is *systemd*'s command line interface utility. The commands can optionally take options, whereas the component whose service you want to check takes the `.service` suffix:

```
$ systemctl <options> <command> <component_name>.service
```

For a complete list of supported commands and options, see the **systemd documentation**

(<https://www.freedesktop.org/software/systemd/man/systemctl.html>).



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

To obtain a list of all running services, run the following command(s):

```
$ systemctl
```

The response is displayed in the following format:

UNIT	LOAD	ACTIVE	SUB	JOB DESCRIPTION
<service_name>	<loaded_or_not>	<active_or_not>	<running_or_not>	
<description_of_the_job>				

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

```
$ systemctl status -l nginx.service
```

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

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

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

```
$ systemctl status -l redis.service
```

To verify if Logstash is running, run the following command(s):

```
$ systemctl status -l logstash.service
```

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

```
$ systemctl status -l elasticsearch.service
```

To retrieve status information about all these services at once, run the following command(s):

```
$ systemctl status -l nginx.service postgresql-9.5.service redis.service  
logstash.service elasticsearch.service
```

To retrieve status information about all the systemd-managed services whose name contains a specific search string, run the following command(s):

```
$ systemctl | grep "<search_string>"
```

Monitor processes

Monitor ingestion queues with Redis

Tool: *Redis* acts as a message broker for Celery-managed tasks.

`redis-cli` is Redis's command line interface utility:

```
$ redis-cli <command> <item_name>
```

For a complete list of supported commands and options, see the **Redis command reference** (<http://redis.io/commands>).

Since Redis's main purpose in this context is to manage task queues, the main and possibly the only command you need is the one that allows you to check queue length: `llen`.

To inspect the platform data ingestion queue length, run the following command(s):

```
$ redis-cli llen "queue:ingestion:inbound"
```

To inspect the graph database queue length, run the following command(s):

```
$ redis-cli llen "queue:graph:inbound"
```

To inspect the Elasticsearch data update queue length, run the following command(s):

```
$ redis-cli llen "queue:search:inbound"
```

Monitor running tasks with Celery

Tool: *Celery* is the task runner that manages task execution and scheduling.

To use Celery to request task information, you need to pass the following environment variables with your request:

```
$ INTELWORKS_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
```

Append Celery commands after the environment variables. Celery commands have the following format:

```
$ celery -A <module_name> <command>
```

Ping Celery to see which tasks are up and listening. This is the easiest way to check task running status. All active tasks reply with `pong`.

```
$ INTELWORKS_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py  
/opt/eclecticiq/platform/api/bin/celery -A eiq.platform.taskrunner.app inspect ping
```

To inspect active tasks, run the following command(s):

```
$ INTELWORKS_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py  
/opt/eclecticiq/platform/api/bin/celery -A eiq.platform.taskrunner.app inspect active
```

To inspect active tasks queues, run the following command(s):

```
$ INTELWORKS_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py  
/opt/eclecticiq/platform/api/bin/celery -A eiq.platform.taskrunner.app inspect  
active_queues
```

To inspect scheduled tasks, run the following command(s):

```
$ INTELWORKS_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py  
/opt/eclecticiq/platform/api/bin/celery -A eiq.platform.taskrunner.app inspect  
scheduled
```

To inspect overall task status, run the following command(s):

```
$ INTELWORKS_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py  
/opt/eclecticiq/platform/api/bin/celery -A eiq.platform.taskrunner.app status
```

To request task statistics (exhaustive, but it can be verbose), run the following command(s):

```
$ INTELWORKS_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py  
/opt/eclecticiq/platform/api/bin/celery -A eiq.platform.taskrunner.app inspect stats
```

(For further details, see the documentation on **Celery ping**

(<http://docs.celeryproject.org/en/latest/userguide/workers.html#ping>), **Celery workers**

(<http://docs.celeryproject.org/en/latest/userguide/workers.html>), **Celery worker statistics**

(<http://docs.celeryproject.org/en/latest/userguide/workers.html#worker-statistics>), and

Celery monitoring (<http://docs.celeryproject.org/en/latest/userguide/monitoring.html>))

Check the Flower database size

Warning:



Whoa! This section is *outdated*!

Flower has been deprecated as the monitoring tool for Celery and Celery tasks.
This section is included as reference.

Flower is a web-based tool to **monitor and manage**

(<https://flower.readthedocs.io/en/latest/features.html>) **Celery**. If Celery tasks seem to lag or hang, you may wish to check the size of the Flower database — a very large database may be a symptom of possible queuing issues:

```
$ ls -lah /opt/eclecticiq/platform/flower.db
```

You can safely delete the Flower database file: if no database exists, Flower automatically creates one at startup or restart.

You can try and delete this database to see if the action frees up some resources:

```
$ rm -f /opt/eclecticiq/platform/flower.db
```

(For further details, see the documentation on **Flower** (<https://flower.readthedocs.io/>))

How to setup Nginx client certificate verification

Set up and configure SSL client certificate verification in Nginx.

Enable client certificate verification in Nginx

Nginx supports client certificate verification through these two configuration options:

- `ssl_client_certificate`
- `ssl_verify_client`

Example:

```
server {  
    listen      443;  
    ssl         on;  
    server_name example.com;  
  
    ...  
  
    ssl_certificate      /etc/nginx/certs/server.crt;  
    ssl_certificate_key  /etc/nginx/certs/server.key;  
    ssl_client_certificate /etc/nginx/certs/ca.crt;  
    ssl_verify_client    on;  
  
    ...  
}
```

The `ca.crt` file is the public key part of a certificate with which the client certificates have been signed. This file should be provided to you by those managing the certificate authority.

Install a client certificate in Google Chrome

To install a client certificate in Google Chrome, do the following:

- Go to **Settings > Show advanced settings.. > HTTPS/SSL > Manage certificates**
- On the **Your certificates** tab, import your client certificate.

To set up a Certificate Authority and generate client and server certificates, we recommend OpenSSL.

How to configure a different database in OpenTAXII

By default, OpenTAXII uses SQLite as a database. You can change this setting to configure a different database, for example PostgreSQL.

OpenTAXII ships with SQLite as its default database. While this is a suitable choice for a development environment where you can test functionality and tinker away to your heart's delight, you may wish to opt for a more powerful solution when you decide to deploy to a production environment that requires added functionality like user management or extensibility and scalability.

PostgreSQL is our recommended database for production. To configure OpenTAXII to work with PostgreSQL, all you need to do is set the `db_connection` parameter of the `opentaxii.yml` configuration file so that it points to a valid PostgreSQL instance. For a connection to a PostgreSQL database, the `db_connection` parameter value observes the following format:

```
db_connection: postgresql://<user_name>:<password>@<db_server>:<port_number>/<db_name>
```

In the following example, the `db_connection` parameter of the `opentaxii.yml` configuration file points to a sample PostgreSQL instance.



All parameter values in the example(s) are dummy values.
Replace these sample values with actual, valid ones before testing them in your environment.

```
domain: "test.taxiistand.com"

auth_api:
  class: opentaxii.auth.sqlldb.SQLDatabaseAPI
  parameters:
    db_connection:
postgresql://username:password@postgresql.server.com:5432/databasename
    create_tables: yes
    secret: som3_s00p3r_s3cr3t_k3y

persistence_api:
  class: opentaxii.persistence.sqlldb.SQLDatabaseAPI
  parameters:
    db_connection:
postgresql://username:password@postgresql.server.com:5432/databasename
    create_tables: yes

logging:
  opentaxii: info
  root: info
```


How to enable audit logging in Kibana

Enable audit logging to examine system events and user access to understand what happened and when, where in the platform, the results it produced, and who/what triggered it.

SysAdmins and SysOps may need to monitor system activities and resource access to troubleshoot issues, investigate a chain of events to identify a root cause, or simply as part of their maintenance routine.

An audit log keeps track of any events that modify the platform configuration or the stored entities. It provides information to identify errors, warnings, or issues affecting specific platform components. It helps you answer these crucial questions:

What?	The running task, action or event (create, delete, update) that produced the outcome you want to investigate.
What?	The modified objects, for example entities or configuration files.
Where?	The affected platform instance (the host system) and the specific area or component, for example incoming feeds, incoming feed tasks, outgoing feed, and so on.
Who?	The user name of the agent that initiated the action.
When?	The time when it occurred (timestamp).

Enable audit logging

By default, audit logging is not configured in the platform.

To enable/disable audit logging, *add* and *set* the following variable to `True` or `False` in the `/opt/eclecticiq/etc/eclecticiq/platform_settings.py` platform configuration file:

Variable	Description
<code>AUDIT_TRAIL_ENABLED = True</code>	Enables audit logging.
<code>AUDIT_TRAIL_ENABLED = False</code>	Disables audit logging.

View audit logs

You can view audit logs in Kibana and in the platform web interface.

View audit logs in Kibana

To view audit logs in Kibana, do the following:

- Sign in to the platform with your user credentials.
- To access Kibana, enter in the web browser address bar a URL with the following format:
`<platform_host_name>/api/kibana/app/kibana#/.`
Keep the trailing `/`.
Example: `https://platform.host.com/api/kibana/app/kibana#/.`
- In Kibana, select the **Discover** view.
- Make sure **logstash-*** is the active index:



- In the search bar, run a search for `logger:eiq.platform.audit_trail`.
- If necessary, adjust the time interval by clicking the clock icon on the top-right corner, and by choosing an appropriate time range for the search.



- If audit logging is enabled, and if the audit log file is populated, the matching audit log records are returned.

▼ March 7th 2016, 16:14:39.854 **logger:** intelworks.platform.audit_trail **body:** {"data": {"entities": ["0c733b12-51b8-487e-bd06-f3e0e0f452b8", "d558eaa7-24e4-4164-b0cf-12bb8e1180c1", "39c7ae02-d9cf-4573-ae36-1712b5e987a8", "472c53f4-fff6-4042-82e6-52821777a494", "27030e5f-95ef-425e-9fa9-2a7379647e2e", "f99ce3fd-0e66-4c6e-a168-fd2fe8855d81"], "extracts": []}}
cookies.platform-api-token: <obfuscated> **headers.accept:** application/json

Table

JSON

[Link to /logstash-2016.03.07/logs/AVNRpPpxCbF3KbRvPeww](#)

@timestamp	🔍 🔍 📄 March 7th 2016, 16:14:39.854
@version	🔍 🔍 📄 1
_id	🔍 🔍 📄 AVNRpPpxCbF3KbRvPeww
_index	🔍 🔍 📄 logstash-2016.03.07
_type	🔍 🔍 📄 logs
body	🔍 🔍 📄 {"data": {"entities": ["0c733b12-51b8-487e-bd06-f3e0e0f452b8", "d558eaa7-24e4-4164-b0cf-12bb8e1180c1", "39c7ae02-d9cf-4573-ae36-1712b5e987a8", "472c53f4-fff6-4042-82e6-52821777a494", "27030e5f-95ef-425e-9fa9-2a7379647e2e", "f99ce3fd-0e66-4c6e-a168-fd2fe8855d81"], "extracts": []}}
cookies.platform-api-token	🔍 🔍 📄 ⚠️ <obfuscated>
headers.accept	🔍 🔍 📄 ⚠️ application/json
headers.accept-encoding	🔍 🔍 📄 ⚠️ gzip, deflate
headers.accept-language	🔍 🔍 📄 ⚠️ en-US,en;q=0.8,nl;q=0.6
headers.authorization	🔍 🔍 📄 ⚠️ <obfuscated>
headers.cache-control	🔍 🔍 📄 ⚠️ no-cache
headers.connection	🔍 🔍 📄 ⚠️ close
headers.content-length	🔍 🔍 📄 ⚠️ 271
headers.content-type	🔍 🔍 📄 ⚠️ application/json

View audit logs in the web interface

To view audit logs in the platform web interface, do the following:

- On the left-hand navigation sidebar click **System**.
- Select the **Audit** tab.
- If audit logging is enabled, and if the audit log file is populated, the matching audit log records are returned. You can sort the table view by column. To do so, click the column header you want to base the data sorting on. An upward-pointing ▲ or a downward-pointing ▼ arrow in the header indicates ascending and descending sort order, respectively.
- You can apply filters to narrow down the search scope:

Filter	Description
Date	Displays only the search result items included in the specified time range.
User	Displays only the search result items with the specified user name(s).
Level	Displays only the search result items with the specified message level flag(s): Info , Warning , Error .

Filter	Description
Method	Displays only the search result items with the specified HTTP method(s): Delete, Post, Put.
Response	Displays only the search result items with the specified HTTP response status code(s): 2xx, 4xx, 5xx.

Dashboard

Discovery

Exposure

beta

Workspaces

Datasets

Tasks

Editor

CONFIGURATION

Incoming Feeds

Outgoing Feeds

Enrichment

Taxonomy

Exclusion List

System

System

USER MANAGEMENT

TAXII

STIX

SERVER

EXPOSURE

LICENSE

AUDIT

Filter...

Date ▾

User ▾

Level ▾

Method ▾

Response ▾

722299 results

DATE	USER	LEVEL	METHOD	RESPONSE	PATH	BODY ▾
2016-03-04 13:45	Bob Test-Admin	INFO	POST	201	/api/utility-tasks/run	{"data": {"is_active": true, "parameters": {"discovery_service_u...
2016-03-04 13:45	Bob Test-Admin	INFO	POST	201	/api/utility-tasks/run	{"data": {"is_active": true, "parameters": {"discovery_service_u...
2016-03-04 13:45	Bob Test-Admin	INFO	POST	201	/api/utility-tasks/run	{"data": {"is_active": true, "parameters": {"discovery_service_u...
2016-03-06 13:51	Bob Test-Admin	INFO	PUT	200	/api/ticket-comments/4	{"data": {"text": "yeah", "ticket": 16}}
2016-03-02 13:42	Bob Test-Admin	INFO	POST	201	/api/taxonomies/	{"data": {"description": "ccc", "name": "xxx"}}



The request payload for an audit log entry cannot exceed 4 KB.

If the request body in the request payload for an audit log entry is larger than 4 KB, the following message is displayed:

Request body too large to log

How to address logging issues in Kibana

Inspect Kibana and Logstash configurations to identify and troubleshoot logging issues.

Prerequisites

This article assumes that the following products are installed and configured in a production and/or in a test environment:

- EclecticIQ Platform
- Elasticsearch
- Logstash
- Kibana

The EclecticIQ Platform runs on CentOS and RHEL; therefore, all command and data output examples apply to these Linux environments.

The ELK stack

Elasticsearch is the search server powering data indexing and retrieval in the EclecticIQ Platform. Elasticsearch is part of the ELK stack, which our platform implements.

The ELK stack puts three products together to provide exceptional insight into your data.

ELK component	Description
Elasticsearch (https://www.elastic.co/products/elasticsearch)	Data indexing and retrieval; analytics.
Logstash (https://www.elastic.co/products/logstash)	Logging, data normalization, data querying.
Kibana (https://www.elastic.co/products/kibana)	Data visualization.

Check Kibana and Logstash configurations to address logging issues

Access Kibana

- Sign in to the platform with your user credentials.
- To access Kibana, enter in the web browser address bar a URL with the following format:
`<platform_host_name>/api/kibana/app/kibana#/.`
Keep the trailing `/`.
Example: `https://platform.host.com/api/kibana/app/kibana#/.`

Check the basics

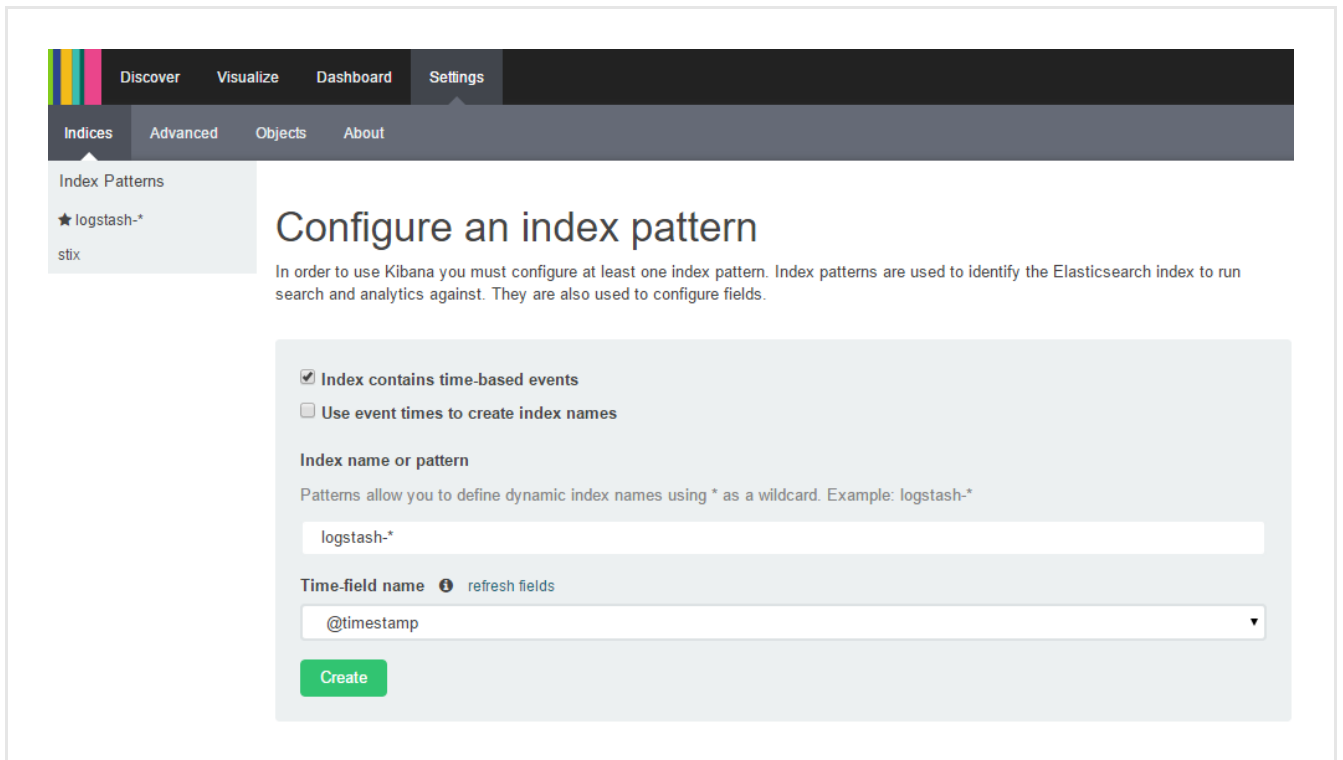
Before starting to dig into data querying, logging and viewing, make sure the basics are set up and configured.

- Make sure the platform dashboard is configured.
- If the Kibana version in use is not exactly the same as the one used to create the dashboard, you need to set the **index pattern** (<https://www.elastic.co/guide/en/kibana/current/tutorial-define-index.html>) for your current Kibana instance.

Create an index

To create an index in Kibana, do the following:

- Sign in to the platform with your user credentials.
- To access Kibana, enter in the web browser address bar a URL with the following format:
`<platform_host_name>/api/kibana/app/kibana#/.`
Keep the trailing `/`.
Example: `https://platform.host.com/api/kibana/app/kibana#/.`
- In Kibana, go to **Settings > Indices**.
- Under **Index Patterns**, click **+ Add New**.
- Under **Configure an index pattern**, create the following index patterns:
 - Set **Index name or pattern** to `logstash-*`.
 - Set **Time-field name** to `@timestamp`.
 - Click **Create**.



Discover Visualize Dashboard **Settings**

Indices Advanced Objects About

Index Patterns

- ★ logstash-*
- stix

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

☒ Index contains time-based events
☐ Use event times to create index names

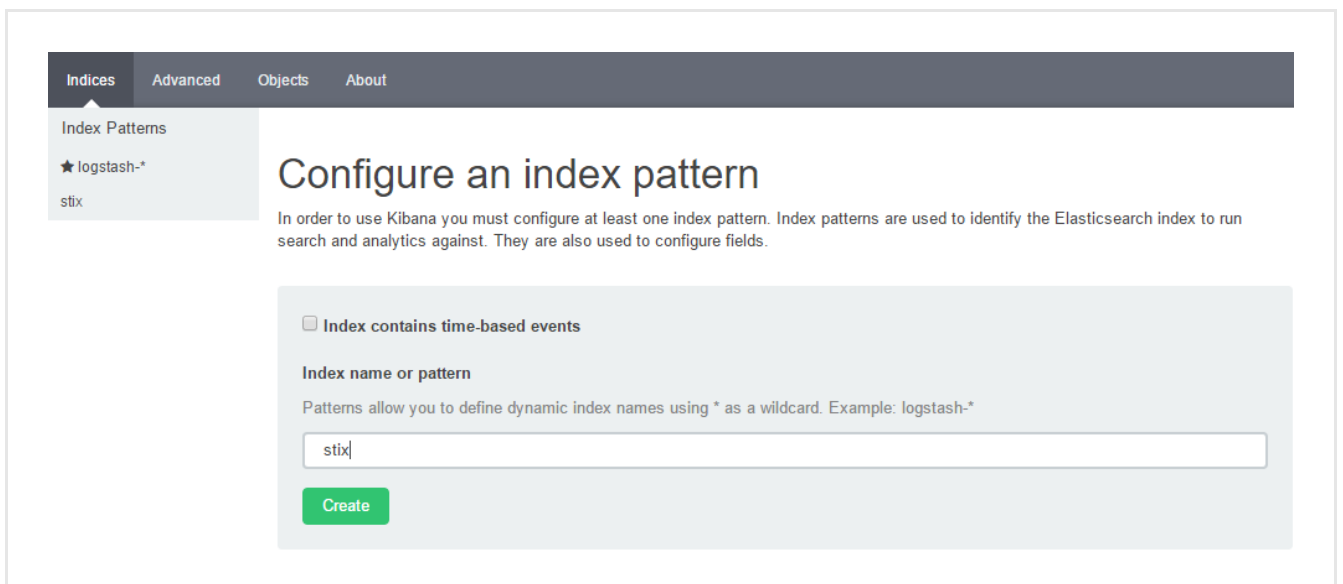
Index name or pattern
 Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

logstash-*

Time-field name ⓘ refresh fields
 @timestamp ▼

Create

- Set **Index name or pattern** to `stix`.
- Deselect the **Index contains time-based events** checkbox.
- Leave **Time-field name** empty.
- Click **Create**.



Indices Advanced Objects About

Index Patterns

- ★ logstash-*
- stix

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

☐ Index contains time-based events

Index name or pattern
 Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

stix

Create

Check Kibana configuration

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 Logstash

Logstash is the component taking care of grunt work like data normalization and log generation. It is not possible to access Logstash directly, but it is important that you check it is running and properly configured. To check if Logstash is running, run the following command(s):

```
$ ps -ef | grep logstash
```

If Logstash is running, any active processes belonging to Logstash are returned.

Check Logstash configuration

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 => ["/var/log/eclecticiq/platform-api.log", "/var/log/nginx/eclecticiq-
platform-api-nginx-access.log", "/var/log/eclecticiq/intel-ingestion.log",
"/var/log/eclecticiq/graph-ingestion.log", "/var/log/eclecticiq/search-ingestion.log"]
    codec => "json"
    tags => "platform-api"
  }
  file {
    path => ["/var/log/eclecticiq/intel-ingestion.log", "/var/log/eclecticiq/graph-
ingestion.log", "/var/log/eclecticiq/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 => ["/var/log/eclecticiq/task-worker-*.log", "/var/log/eclecticiq/task-
beat.log", "/var/log/eclecticiq/task-queue-analysis-listener.log",
"/var/log/eclecticiq/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.

To perform several tasks in the procedure, you may need root-level access rights.

To obtain administration rights, run the following command(s):

```
$ sudo su -
```

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

```
$ 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 Logstash 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 transformations.
- `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.html#encodes>) before outputting it to the shell standard output. You may edit it as needed.

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

Test Logstash configuration

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

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

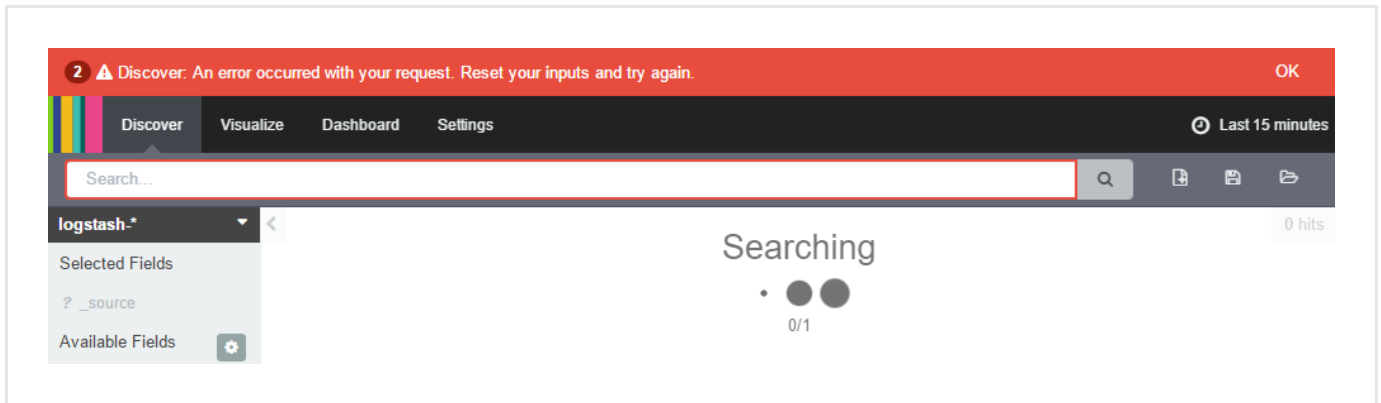
- If the configuration test result is positive, the response reads:

```
Configuration OK
```

Go back to Kibana

After inspecting the Kibana and Logstash configurations and editing them, if necessary, you can go back to Kibana, and you can select **Discover** to view all log data recorded in the last 15 minutes.

It may happen that you are denied access to the Kibana **Discover** view:

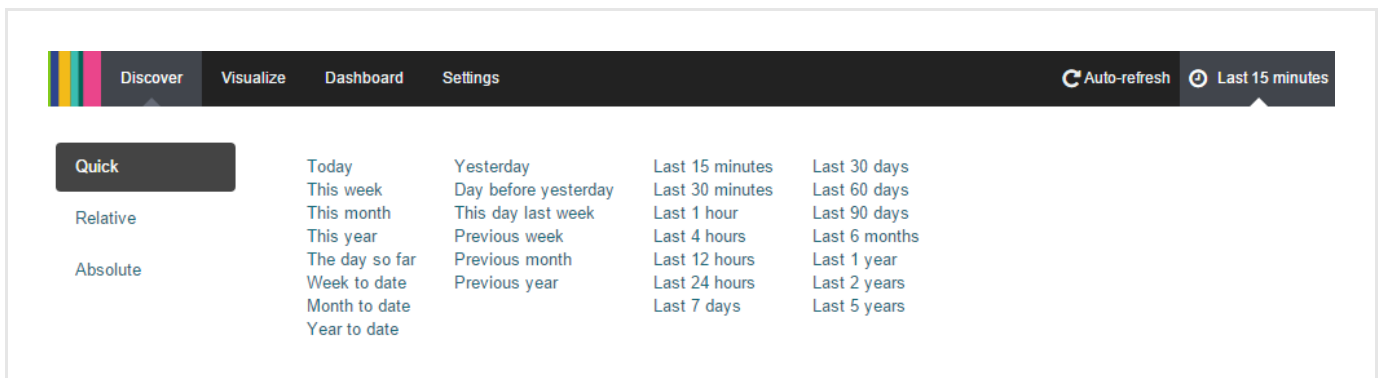


This is usually due to an access timeout. To log back in to Kibana, do the following:

- Go back to the platform login page.
- Sign in to the platform with your user credentials.
- To access Kibana, enter in the web browser address bar a URL with the following format:
`<platform_host_name>/api/kibana/app/kibana#/.`
 Keep the trailing `.`
 Example: `https://platform.host.com/api/kibana/app/kibana#/.`

Adjust the update interval

By default, the log data update interval is set to 15 minutes. You can adjust it to match your requirements by clicking it, and then by choosing the appropriate value among the available options.



Configure output to syslog

You can use this plugin to send Logstash logs to syslog to be output: **Logstash output syslog plugin**

(<https://www.elastic.co/guide/en/logstash/current/plugins-outputs-syslog.html>).

How to search logs for issues in Kibana

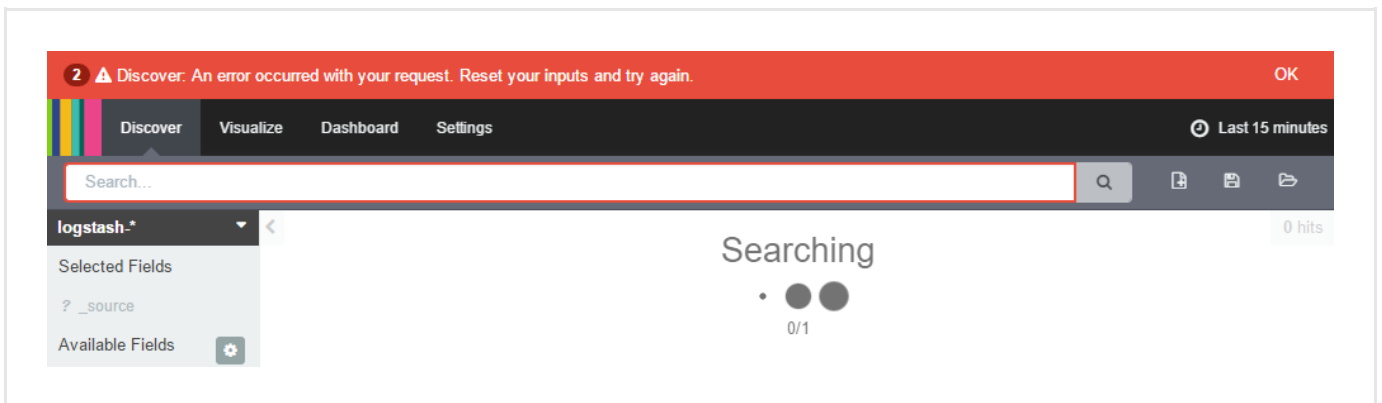
Search Kibana to retrieve log data about errors, warnings, or issues concerning specific platform components.

Search Kibana to retrieve log data about issues

Access Kibana

- Sign in to the platform with your user credentials.
- To access Kibana, enter in the web browser address bar a URL with the following format:
`<platform_host_name>/api/kibana/app/kibana#/.`
Keep the trailing `.`
Example: `https://platform.host.com/api/kibana/app/kibana#/.`

It may happen that you are denied access to the Kibana **Discover** view:

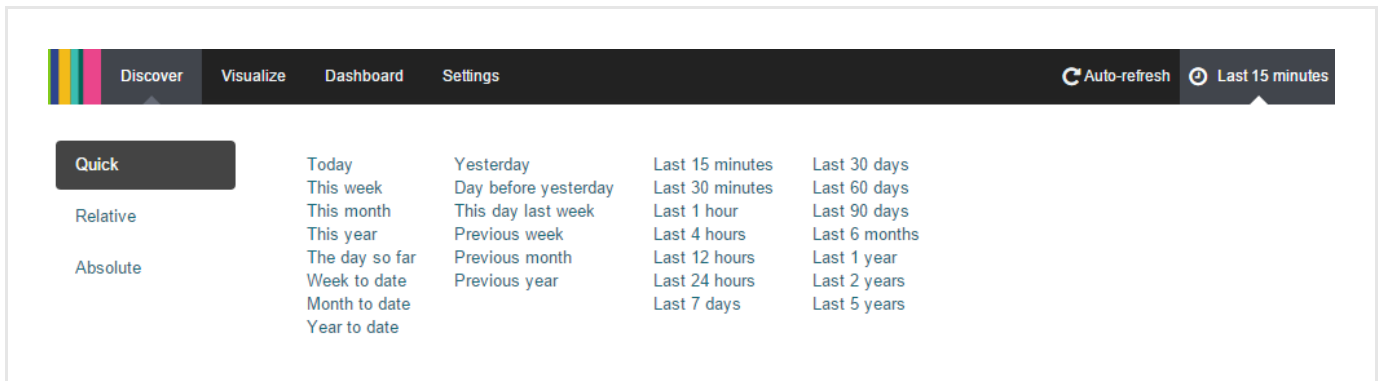


This is usually due to an access timeout. To log back in to Kibana, do the following:

- Go back to the platform login page.
- Sign in to the platform with your user credentials.
- To access Kibana, enter in the web browser address bar a URL with the following format:
`<platform_host_name>/api/kibana/app/kibana#/.`
Keep the trailing `.`
Example: `https://platform.host.com/api/kibana/app/kibana#/.`

Adjust the update interval

By default, the log data update interval is set to 15 minutes. You can adjust it to match your requirements by clicking it, and then by choosing the appropriate value among the available options.



Search by level

Log records are assigned a severity level. In this way, you can filter searches to retrieve only errors, warnings, or informative/notification log records.

To run a search by level in Kibana, do the following:

- In Kibana, select the **Discover** view.
- Make sure **logstash-*** is the active index:



Enter this in the Kibana search bar...	...to obtain this result
<code>level:"error"</code>	Returns all logging errors.
<code>level:"warning"</code>	Returns all logging warnings.
<code>level:"info"</code>	Returns all logging information and notifications.

Search by tag

Log records are tagged, so that you can easily identify the component a log refers to. To run a search by tag in Kibana to look for issues or information about a specific component, do the following:

- In Kibana, select the **Discover** view.
- Make sure **logstash-*** is the active index:



In the Kibana search bar enter this...	...to get this result
<code>tags.raw:"ingestion"</code>	Returns logs related to the intel ingestion module controlling how the platform ingests source data.
<code>tags.raw:"opentaxii"</code>	Returns logs related to the EclecticIQ OpenTAXII server implementing the TAXII services.
<code>tags.raw:"platform-api"</code>	Returns logs related to the EclecticIQ Platform API.
<code>tags.raw:"task-workers"</code>	Returns logs related to the services and processes carrying out complementary tasks.

Search using Boolean operators

You can use Boolean operators to refine your search and zero in on specific log types or components. For example you can search for Redis and/or Neo4j issues by entering in the Kibana search bar the following query:

```
level:"error" AND tags.raw:"graph-ingestion"
```

You can search for Redis and/or Neo4j issues, or for issues concerning how data is ingested, by entering in the Kibana search bar the following query:

```
level:"error" AND tags.raw:"graph-ingestion" OR tags.raw:"intel-ingestion"
```

How to reindex Elasticsearch

You may need to reindex Elasticsearch for several reasons: from changes to data types or data analysis, to updating the Elasticsearch schema by adding or removing fields. Whenever a change in the data structure is introduced, you should reindex Elasticsearch to make sure existing and new data is correctly indexed and stored in the Elasticsearch database.

Reindex Elasticsearch



Depending on the size of the Elasticsearch database and the workload on the system, you may want to consider stopping some non-core platform processes — for example ingestion queues — before reindexing to free up system resources.

- To reindex Elasticsearch, you first need to define `elasticsearch` as the designated superuser to execute the reindexing command with. Run the following command(s):

```
$ sudo -u elasticsearch -i
```

- To launch the Elasticsearch database reindexing, run the following command(s):

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

Fully 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.

```
$ INTELWORKS_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  
}
```

How to reindex the graph database

You may need to reindex the graph database for several reasons: from changes to data types or data analysis, to updating the data schema by adding or removing fields. Whenever a change in the data structure is introduced, you should reindex the graph database to make sure existing and new data is correctly indexed and stored.



Depending on the size of your database, *reindexing the graph database can take from several hours to a few days.*

It is a good idea to run `reindex_graph` in a separate screen session, so that you can monitor the process.

The `reindex_graph` command adds and enqueues entities, enrichments, and observables stored in the platform to the graph ingestion queue:

```
$ /opt/eclecticiq/platform/api/bin/manage reindex_graph -o <offset_value> -q <queue_name>
```

Example:

```
$ /opt/eclecticiq/platform/api/bin/manage reindex_graph -o 0 -q queue:graph:inbound
```

Parameter	Description
-?	Shows the built-in help.
-i <item_type>	Ex.: <code>entity</code> . Allowed values/Supported types: <code>entity</code> , <code>enrichment</code> , <code>extract</code> . When no item type is specified, the command reindexes all graph items of all supported item types.
-q <queue_name>	Ex.: <code>queue:graph:inbound</code> . The queue to reindex and ingest again. If no queue is specified, the default queue for the process is the graph ingestion queue: <code>queue:graph:inbound</code> .
-o <offset_value>	Ex.: <code>-o 10</code> . Offset value, integer. Default value: 0. Typically, graph reingestion and reindexing selects all the entities IDs stored in the platform, sorts them in ascending order by <code>created_at</code> , and then adds all the IDs to the graph ingestion queue. If an offset value is defined, the process ignores the first <i>n</i> entities corresponding to the specified offset.

Parameter	Description
<code>-l <limit_value></code>	Ex.: <code>-l 200</code> . Capping value, integer. Default value: <code>100000</code> . Typically, graph reingestion and reindexing selects all the entities IDs stored in the platform, sorts them in ascending order by <code>created_at</code> , and then adds all the IDs to the graph ingestion queue. If a limit value is defined, only the first n entities corresponding to the specified limit are processed.

How to back up and restore a PostgreSQL database

Back up and restore a PostgreSQL database, for example after upgrading or reinstalling the platform, or as a disaster recovery mitigation strategy.

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

Back up the PostgreSQL database

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
```

Restore the backup

To restore backed up data, follow the standard recommendations and procedures for PostgreSQL:

- **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>)

Check, check, check

Check the PostgreSQL configuration

If you upgraded PostgreSQL along with the platform, make sure the port assigned to PostgreSQL has not changed, compared to the previous version:

Go to the `*/var/lib/pgsql//data/*` to review the `*postgresql.conf*` configuration file:

```
$ sudo -e cd /var/lib/pgsql/<version_number>/data/  
  
$ nano postgresql.conf
```

Make sure that the `port` value has not changed between versions. By default, PostgreSQL uses port 5432:

```
listen_addresses = '*'
port = 5432
max_connections = 100
log_destination = 'stderr'
logging_collector = on
log_directory = '/var/log/postgresql'
log_filename = 'postgresql-%Y-%m-%d.log'
log_truncate_on_rotation = on
log_rotation_age = 1d
log_rotation_size = 0
log_line_prefix = '%m '
log_timezone = 'UTC'
datestyle = 'iso, mdy'
timezone = 'UTC'
lc_messages = 'C'
lc_monetary = 'C'
lc_numeric = 'C'
lc_time = 'C'
default_text_search_config = 'pg_catalog.english'

# Increase allowed memory usage.
shared_buffers = 1024MB
work_mem = 8MB

# Random page lookups are cheap on SSD disks.
random_page_cost = 1.5

# Maintenance operations are usually not performed concurrently, so
# they can use significantly more memory to speed things up.
maintenance_work_mem = 1GB
```

If you upgraded PostgreSQL from a previous version to a newer one, and then restored the database SQL dump backup copy, you can remove the previous PostgreSQL version.

Check the platform configuration

To make sure the platform sees the upgraded PostgreSQL database, verify that the `SQLALCHEMY_DATABASE_URI` parameter in the `platform_settings.py` platform configuration file points to the correct database and to the correct address/port – the `<db_password>` parameter in the URI should be pre-populated:

```
$ cd opt/eclecticiq/etc/eclecticiq/

$ nano platform_settings.py
```

```
SQLALCHEMY_DATABASE_URI = "postgresql://platform-api:  
<db_password>@localhost:5432/platform-api"
```

Check the PostgreSQL service

Now you can launch the platform and check if the systemd-managed services are running to verify that the upgraded PostgreSQL is up and running as well. If the upgraded version uses the same port as the previous one, the platform should pick it up when it launches.

You can check if the correct version is up and running:

```
$ systemctl status | grep "<version_number>"
```

Alternatively, you can check if the `postgres` service is up and running. This command returns a more verbose response:

```
$ systemctl status | grep "postgres"
```

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 as 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.

How to shut down the platform

Graciously shut down the platform by first stopping its core services and processes.

You may need to stop the platform and its services, for example if you run out of disk space and need to add more storage, or before applying an upgrade.

To gracefully shut down EclecticIQ Platform, do the following:

- Stop all running processes.
- Shut down the machine the platform is hosted on.

Generic shutdown

You can stop the platform without following any specific procedure. However, if you want to make sure the platform core services and processes gracefully shut down, manually execute the following commands:

```
$ supervisorctl stop all
```

```
$ systemctl stop postgresql-9.5 redis logstash elasticsearch
```

You can then proceed to shut down the virtual or physical machine hosting the platform.

Shutdown before a platform upgrade

If you are shutting down the platform before performing an upgrade, stop platform components in the order described below to make sure no Celery tasks are left over in the queue. This prevents hanging tasks in the queue from interfering with the upgrade procedure.

- Stop `platform-api`:

```
$ supervisorctl stop platform-api
```

- Stop the Celery beat:

```
$ supervisorctl stop task:beat
```

- Check Celery queues — they should be empty:

```
$ redis-cli llen enrichers  
  
$ redis-cli llen integrations  
  
$ redis-cli llen priority_enrichers  
  
$ redis-cli llen priority_providers  
  
$ redis-cli llen priority_utilities  
  
$ redis-cli llen providers  
  
$ redis-cli llen reindexing  
  
$ redis-cli llen utilities
```

Alternatively, check Redis keys:

```
$ redis-cli keys *
```

If the response *does not include* any keys related to any Celery queues, they are empty and you can continue.

- Stop the remaining Celery workers:

```
$ supervisorctl stop task:*
```

- Stop Supervisor-managed workers:

- kibana
- intel-ingestion
- opentaxii
- search-ingestion
- graph-ingestion
- neo4j

```
$ supervisorctl stop all
```

Alternatively, you can list them explicitly:

```
$ supervisorctl stop kibana intel-ingestion opentaxii search-ingestion graph-ingestion  
neo4j
```

