Install and configure EclecticIQ Platform

Install and config guide for system administrators

Last generated: March 06, 2018

Table of contents

EclecticIQ Platform installation and configuration — CentOS / RHEL

This document guides you through the installation, setup and configuration of EclecticIQ Platform when you choose to install the product from an RPM package or a tarball 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:

- 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 or a tarball.

Command examples

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 `$`, and they look like this:

```
$ run command
```

Code examples

Inline code comments, especially for bash and shell command examples, are usually prefixed by `#`:

```
# This is a self-explanatory inline code comment

$ run command
```

Code and configuration examples look like this:

```
{
  "this" : [
    {
      "key_name" : "key_value"
    }
  ]
}
```

The actual format of each code snippet depends on the corresponding programming language.

Text editors

When you need to open a file to edit it, code examples use the Vim text editor.
Example:

```
$ vi /etc/nginx/conf.d/platform.conf
```

Feel free to replace *vi* with your preferred weapon of choice, be it ***Vim*** `(http://www.vim.org/)`, ***Emacs*****, or** `(https://www.gnu.org/software/emacs/)`***Nano*** `(https://www.nano-editor.org/)`**.**
`(https://imgs.xkcd.com/comics/real_programmers.png)`

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, in many different formats.
Enrichment	Enrich existing intelligence with external data sources, and refine it with de-duplication and pattern recognition.
Sharing	Investigate and identify threats together with partners as part of an information ecosystem.
Collaboration	Analyze and create intelligence in collaboration with other teams and departments.
Insights	Generate insight thanks to a high-fidelity, normalized view into your intelligence.
Integration	Understand how cyber intelligence relates to and how it can affect your organization and your environment.

Hardware requirements

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

Single box

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

HW area	Minimum	Recommended	Notes
Environment	-	Physical machine/*rpm* and *deb* installs	
CPUs	4	8	Core count includes HT
CPU speed	2.5 GHz	2.5 GHz or faster	
Memory	32 GB	64 GB or more	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. Operations and tasks carried out through the web-based UI may be memory-intensive: the web browser can use ~1 GB or more, occasionally. 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

HW area	Minimum	Recommended	Notes
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
	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

Data mount points

When installing and configuring platform components such as PostgreSQL, Elasticsearch, and Neo4j you need to specify dedicated locations where these products store their data.

The table below shows the recommended mount point paths for each data store.

Component	Mount point	Minimum size (GB)	Recommended size (GB)
Logs	`/var/log`	10	20
Neo4j	`/media/neo4j`	25	50
Elasticsearch	`/media/elasticsearch`	50	75
PostgreSQL	`/media/pgsql`	200	300

Redis is installed to the root partition where the platform is installed to.
During the configuration step, you can set the Redis data location in the *redis.conf* configuration file. The recommended target directory for Redis data is */media/redis*.
This is not a mount point on a separate partition, it is a subdirectory in the root partition.

Scaling out

The easiest approach to scaling out involves 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 dependencies and third-party products, ensure you have the following information available:

- DNS name of the host machine 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 name	`packer`
password	`Packer123!`

EclecticIQ Platform default login credentials	
user name	`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/)
- **Ubuntu Server 16.04 LTS (Xenial Xerus)** (https://wiki.ubuntu.com/xenialxerus/releasenotes)

Web browsers

EclecticIQ Platform web-based GUI supports the following web browsers:

- Google Chrome (latest)

Encoding

The platform default character encoding is **UTF-8** (http://www.utf-8.com/). Dependencies and components that exchange data with the platform need to apply the same encoding.

Time zone

The global time zone configuration needs to be **UTC**.

> While you can set a local or a custom time zone value for the platform, the host environment needs to be consistently on **UTC time**.
> This includes OS, databases, as well as any other products or components that allow setting a time zone, and that interact/interoperate with the platform.

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.

Dependency	Version	Reference
Oracle Java JDK	1.8.0	**Oracle Java SE web site** (https://www.oracle.com/java/technologies/java-se.html)
PostgreSQL	10.1	**PostgreSQL web site** (https://yum.postgresql.org/repopackages.php)
Redis	3.2.10	**Redis web site** (http://redis.io)
Nginx	1.12.x	**Nginx web site** (https://nginx.org)
Neo4j	3.3.3 Community	**Neo4j web site** (http://neo4j.com)
Elasticsearch	5.6.7	**Elasticsearch web site** (https://www.elastic.co)
elasticdump	3.0.0	**elasticdump Node js module page** (https://www.npmjs.com/package/elasticdump)
Logstash	5.6.8	**Logstash reference documentation** (https://www.elastic.co/guide/en/logstash/5.6/index.html)
Kibana	5.6.8	**Kibana reference documentation** (https://www.elastic.co/guide/en/kibana/current/getting-started.html)
unzip	n/a	**Compress and decompress file archives** ()
Node.js	6.x.x or later	**Node.js web site** (https://nodejs.org/)
poppler-utils	n/a	**poppler-utils reference page** (https://apps.fedoraproject.org/packages/poppler-utils)
Postfix	n/a	**Postfix web site** (http://www.postfix.org/)
Supervisor	n/a	**Supervisor web site** (http://supervisord.org/)
StatsD	n/a	**Metrics aggregator for the dashboard** (https://github.com/etsy/statsd)
statsd-elasticsearch-backend	n/a	**Backend for Elasticsearch to work with StatsD** (https://github.com/markkimsal/statsd-elasticsearch-backend)

Warning: About Elasticsearch

During complex index upgrades and reindexing operations, Elasticsearch may require additional disk space to store temporary working files and temporary copies of the existing indices.

Monitor your Elasticsearch partition usage. Before it reaches 50% of the available space in the partition, extend it, so that the new partition size is at least twice as large as the sum of the existing Elasticsearch indices.

Example: if Elasticsearch currently uses 43 GB of disk space, extend the partition where Elasticsearch lives, so that it is at least 86 GB.

Bundled third party software

EclecticIQ Platform is bundled with the following third party software.
Each product on the list abides by its own terms and conditions and its own license.

Dependency	Version	Reference
alembic	0.9.6	Bitbucket
amqp	1.4.9	GitHub
anyjson	0.3.3	Bitbucket
apispec	0.4.1	GitHub
appnope	0.1.0	GitHub
argskwargs	1.0.3	GitHub
asn1crypto	0.23.0	GitHub
attrs	17.3.0	attrs
bcrypt	3.1.4	GitHub
beautifulsoup4	4.6.0	Crummy
billiard	3.3.0.23	GitHub
blinker	1.4	PythonHosted
boto3	1.4.7	GitHub
botocore	1.7.45	GitHub
cabby	0.1.18	GitHub
cachetools	2.0.1	GitHub
celery	3.1.18	Celery
certifi	2017.11.5	Certifi
cffi	1.11.2	CFFI
chardet	3.0.4	GitHub
click	6.7	GitHub
colorama	0.3.9	GitHub
colorlog	3.1.0	GitHub
cryptography	2.1.3	GitHub
datauri	1.0.0	GitHub
decorator	4.1.2	GitHub

Dependency	Version	Reference
defusedxml	0.5.0	GitHub
docutils	0.14	Docutils
elasticsearch	1.7.0	GitHub
fancycompleter	0.8	Bitbucket
feedparser	5.2.1	GitHub
flamegraph	0.1	GitHub
Flask	0.10.1	GitHub
Flask-Classy	0.6.10	GitHub
Flask-JWT	0.2.0	GitHub
flask-marshmallow	0.8.0	GitHub
Flask-Redis	0.3.0	GitHub
Flask-SQLAlchemy	2.3.2	GitHub
furl	1.0.1	GitHub
future	0.16.0	Python-Future
gunicorn	19.7.1	Gunicorn
idna	2.6	GitHub
inflect	0.2.5	pyPI
ipdb	0.10.3	GitHub
ipython	6.2.1	IPython
ipython_genutils	0.2.0	IPython
itsdangerous	0.24	GitHub
jedi	0.11.0	GitHub
Jinja2	2.10	Jinja
jmespath	0.9.3	GitHub
kombu	3.0.37	Kombu
libtaxii	1.1.111	TAXII
lxml	4.1.1	lxml
Mako	1.0.7	mako
MarkupSafe	1.0	GitHub
marshmallow	2.6.1	GitHub

Dependency	Version	Reference
marshmallow-sqlalchemy	0.13.2	GitHub
newrelic	2.96.0.80	New Relic
orderedmultidict	0.7.11	GitHub
paramiko	2.4.0	GitHub
parso	0.1.0	GitHub
pdbpp	0.9.2	GitHub
pexpect	4.3.0	Pexpect
pickleshare	0.7.4	GitHub
prompt_toolkit	1.0.15	GitHub
psycopg2	2.7.3.2	init.d
ptyprocess	0.5.2	GitHub
pyasn1	0.3.7	GitHub
pycparser	2.18	GitHub
Pygments	2.2.0	Pygments
PyJWT	1.4.0	GitHub
pyldap	2.4.25.1	GitHub
pymisp	2.4.80	GitHub
PyNaCl	1.2.0	GitHub
pyOpenSSL	17.3.0	pyOpenSSL
pysaml2	4.5.0	GitHub
python-dateutil	2.4.2	dateutil
python-editor	1.0.3	GitHub
python-gnupg	0.4.1	PythonHosted
python-magic	0.4.13	GitHub
python-slugify	1.2.4	GitHub
pytz	2017.3	PythonHosted
PyYAML	3.12	PyYAML
rarfile	3.0.0	GitHub
redis	2.10.6	GitHub

Dependency	Version	Reference
requests	2.18.4	Requests
requests-futures	0.9.7	GitHub
retrying	1.3.3	GitHub
rfc3986	1.1.0	RFC 3986
s3transfer	0.1.11	GitHub
simplegeneric	0.8.1	CheeseShop
six	1.11.0	PyPI
SQLAlchemy	1.1.15	SQLAlchemy
statsd	3.2.1	GitHub
structlog	16.0.0	structlog
tld	0.7.9	GitHub
traitlets	4.3.2	IPython
unrar	0.3	GitHub
urllib3	1.22	urllib3
wcwidth	0.1.7	GitHub
Werkzeug	0.12.2	The Pallets Projects
wmctrl	0.3	Bitbucket

SELinux

EclecticIQ Platform supports **SELinux** `(http://selinuxproject.org/)`.

- If you are using or plan to use SELinux in the environment where the platform is installed, you should carry out this 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.

Check SELinux status

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
```

Check SELinux mode

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.

Set SELinux to permissive mode

The recommended SELinux mode to offload complexity during installation and configuration is `permissive`.
To set SELinux to work permissively, run the following command(s):

```
$ setenforce permissive
```

Post-installation check

The platform post-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 is disabled, the post-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 it is enabled, and if the platform post-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"
```

- 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.

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.

Install dependencies

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

- Required credentials for installation:

```
eiq_user=
eiq_pass=
```

- Required variables for installation:

```
platform_deps_path=downloads.eclecticiq.com/platform-centos-deps/2.1
binary_repo_path=downloads.eclecticiq.com/platform-binary-deps
```

Credentials and variables listed above need to be exported prior to continuing with the instructions: `export var_name=value` or they can be replaced with their values in any of the commands that contain them.

Keep root-level access rights handy to successfully execute multiple commands using the command line or the terminal.

To obtain admin rights, run the following command(s): `sudo su -`

Alternatively:

- Grant admin rights to a specific user, who can then log in with their password to perform admin tasks: `su - USER_NAME`

Create a YUM repository for the dependencies

You can download the core dependencies for the platform from a centralized YUM repository. This makes it easier for system administrators to manage platform installation and upgrade procedures, and it ensures that the platform dependency versions match the requirements.

This YUM repository provides the following platform dependencies:

- Elasticsearch 5.6.0
- Kibana 5.6.0
- Logstash 5.6.0
- Neo4j 3.3.0
- Nginx 1.12.0
- Poppler-Utils 0.26.5
- Postfix 2.10.1
- PostgreSQL 10.1
- Python 3.6.3
- Redis 3.2.3
- StatsD 0.7.2
- Supervisor 3.1.4

Additional dependencies may be collected as binary packages from `https://downloads.eclecticiq.com/platform-binary-deps/`:

- Elasticdump 3.0.0
- JavaJDK 8u151
- Node.js 6.11.4, including npm

Create the YUM repository configuration file

- Create a new file with the repository information:

```
# Create and populate the YUM repo file for the platform dependencies
echo "
[platform-centos-deps]
name=platform-centos-deps
baseurl=https://${platform_deps_path}
gpgcheck=0
repo_gpgcheck=0
enabled=1
username=${eiq_user}
password=${eiq_pass}
gpgkey=https://downloads.eclecticiq.com/public/GPG-KEY-eclecticiq
       https://downloads.eclecticiq.com/public/GPG-KEY-elasticsearch
       https://downloads.eclecticiq.com/public/neo4j.gpg.key
       https://downloads.eclecticiq.com/public/nginx_signing.key
       https://downloads.eclecticiq.com/public/IUS-COMMUNITY-GPG-KEY
       https://downloads.eclecticiq.com/public/RPM-GPG-KEY-EPEL-7
       https://downloads.eclecticiq.com/public/NODESOURCE-GPG-SIGNING-KEY-EL.key
priority=1
" > /etc/yum.repos.d/platform-centos-deps.repo
```

Update and set user name and password as needed, based on your system environment.

Install the core dependencies

- Install the following core dependencies:

```
# Install core dependencies
yum -y --disablerepo=* --enablerepo="platform-centos-deps,epel,base" install epel-release
supervisor-3.1.4* nginx-1.12.0* redis-3.2.3* elasticsearch-5.6.0* logstash-5.6.0* kibana-5.6.0*
neo4j-3.3.0* python36u-3.6.3 yum-plugin-versionlock
```

- Install version lock plugin for yum:

```
Install versionlock
yum -y install yum-versionlock
```

- Pin the following packages so that they stay on the corresponding current versions and are not automatically updated:

```
# Pin package versions
yum versionlock elasticsearch logstash kibana neo4j python36u
```

Install the other required dependencies

PostgreSQL

- Install PostgreSQL 10 client and repo keys:

```
# Install PostgreSQL client and keys
yum -y --disablerepo=* --enablerepo="platform-centos-deps,epel,base" install postgresql10-10.1
```

- Install PostgreSQL server and additional components by running the following command(s):

```
# Install PostgreSQL server
yum -y --disablerepo=* --enablerepo="platform-centos-deps,epel,base" install postgresql10-devel
postgresql10-contrib postgresql10-server postgresql10-libs
```

- Pin the following packages, so that they stay on the corresponding current versions, and they are not automatically updated:

```
# Pin package versions
yum versionlock postgresql10-server postgresql10-contrib postgresql10-devel
```

Node.js

- Remove any previous Node.js and npm installations

```
# Remove nodejs
yum -y remove nodejs* npm
```

- Retrieve and run a script that installs and enables Node.js repo:

```
# Retrieve Node.js
wget --user "${eiq_user}" --password "${eiq_pass}" -qO-
https://${binary_repo_path}/node_rpm_setup_6.x.sh | bash -
```

- Install Node.js 6.x:

```
# Install Node.js
yum -y --disablerepo=* --enablerepo="platform-centos-deps,epel,base" install nodejs
```

Statsd

- Install statsd. Node.js needs to be installed prior to statsd!

```
# Install statsd
wget --user "${eiq_user}" --password "${eiq_pass}" -qO- https://${binary_repo_path}/statsd-
0.7.2.tar.gz | tar -xzf - -C /opt/
```

Elasticsearch plugins

- Install elasticdump 3.0.0 with this command:

```
# Install elasticdump globally
escaped_username=$(echo $eiq_user | sed 's|\@|%40|g')
npm install --username -g https://${escaped_username}:${eiq_pass}@${binary_repo_path}/elasticdump-
3.0.0.tgz
```

Neo4j

- Setup a profile file for Neo4j where you define the environment variables:

```
# Export profile for Neo4j
echo 'export NEO4J_HOME=/usr/share/neo4j
export PATH=$PATH:$NEO4J_HOME/bin' > /etc/profile.d/neo4j.sh
```

- Make sure these environment variables are accessible to the neo4j user, neo4j group profile by checking the output of `su -s /bin/bash neo4j -c 'echo $NEO4J_HOME; echo $PATH'` which should look like this:

```
/usr/share/neo4j
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin:/usr/share/neo4j
```

Postfix

- Install postfix:

```
# Install postfix
yum -y --disablerepo=* --enablerepo="platform-centos-deps,epel,base" install postfix-2.10.1
```

Java SE Development Kit

Obtain Oracle Java SE Development Kit 8 by carrying out the following steps:

- Download Oracle Java JDK 8 by replacing the ${eiq_user} and ${eiq_pass} placeholders in the URL with your EclecticIQ login credentials:

```
# Install Oracle Java JDK 8
wget --user "${eiq_user}" --password "${eiq_pass}" -qO- https://${binary_repo_path}/jdk-8u151-
linux-x64.tar.gz | tar -xzf - -C /opt/
```

- Set Oracle Java JDK as default Java by running the following command(s):

```
# Set Oracle Java JDK as default
update-alternatives --install /usr/bin/java java /opt/jdk1.8.0_151/bin/java 100
update-alternatives --install /usr/bin/javac javac /opt/jdk1.8.0_151/bin/javac 100
update-alternatives --set java /opt/jdk1.8.0_151/bin/java
update-alternatives --set javac /opt/jdk1.8.0_151/bin/javac
```

- Verify the JDK version installed on the target system: `java -version` with expected output:

```
java version "1.8.0_151"
Java(TM) SE Runtime Environment (build 1.8.0_151-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)
```

- Set up a profile file for JDK where you define the JAVA_HOME environment variable:

```
# Export Java home
echo 'export JAVA_HOME=/opt/jdk1.8.0_151' > /etc/profile.d/jdk.sh
```

poppler-utils

The platform requires poppler-utils to process PDF files.

- Install poppler-utils by running the following command(s):

```
# Install poppler-utils
yum -y --disablerepo=* --enablerepo="platform-centos-deps,epel,base" install poppler-utils
```

Optional tools

- JQ is used by our install scripts. Install jq utility if it's not present:

```
# Install jq
wget --user "${eiq_user}" --password "${eiq_pass}" -qO- https://${binary_repo_path}/jq-1.5.tar.gz
| tar -xzf - -C /usr/bin/
```

- Rsync is used by our install scripts. Install jq utility if it's not present:

```
# Install rsync
yum -y install rsync
```

Install from an rpm package

After checking the necessary requirements and dependencies the platform needs to work, you can proceed to install the platform from a YUM repository and an rpm package.

- Required credentials for installation:

```
eiq_user=
eiq_pass=
```

- Required variables for installation:

```
platform_repo_path=downloads.eclecticiq.com/platform-centos/2.1
```

Credentials and variables listed above need to be exported prior to continuing with the instructions: `export var_name=value` or they can be replaced with their values in any of the commands that contain them.

Install from the YUM repository

Create the YUM repository for the platform

If you are upgrading or performing a fresh install of the EclecticIQ Platform using the YUM package manager, you should define your YUM repository configuration for the platform. To do so, create and populate `/etc/yum.repos.d/eclecticiq-platform.repo` file by running the following commands and replacing `${eiq_user}` and `${eiq_pass}` with proper values:

```
# Create and populate yum repo file for EclecticIQ Platform
echo "
[platform-centos]
name=platform-centos
baseurl=https://${platform_repo_path}
gpgcheck=1
repo_gpgcheck=0
enabled=1
username=${eiq_user}
password=${eiq_pass}
gpgkey=https://downloads.eclecticiq.com/public/GPG-KEY-eclecticiq
" > /etc/yum.repos.d/platform-centos.repo
```

- The installation or upgrade procedure should normally take care of removing `/etc/yum.repos.d/private-eclecticiq.repo`. In case it is not removed, you can remove it manually using: `rm -f /etc/yum.repos.d/private-eclecticiq.repo`.

Install the platform

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

```
# Install EclecticIQ Platform packages
yum -y install 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 assets and resources.

The platform documentation is shipped together with the platform installation packages. When you install or upgrade the platform, the documentation is included in the process.

However, you may occasionally wish to install the documentation separately e.g. updating help to a recent version.

To install the latest platform documentation on CentOS and RHEL so that it is available in-product as a help resource, do the following:

```
# Install platform documentation
yum -y install eclecticiq-platform-docs
```

Check permissions

Supervisor configuration and log file locations should be owned by `eclecticiq:eclecticiq`:

```
# Be sure about proper ownership of configuration and log file locations
chown -R eclecticiq:eclecticiq /var/run/supervisor/
chown -R eclecticiq:eclecticiq /etc/supervisord.d/
chown -R eclecticiq:eclecticiq /var/log/supervisor/
chown -R eclecticiq:eclecticiq /var/log/eclecticiq/
```

If you reboot the system after applying these changes, ownership of the /var/run/supervisor directory may be reset to `root:root`. To address this issue, create a supervisor.conf file in the /etc/tmpfiles.d directory:

```
# Make sure /var/run/supervisor has proper ownership on every boot
echo 'D /var/run/supervisor 0775 eclecticiq eclecticiq -' > /etc/tmpfiles.d/supervisor.conf
```

Post-installation configuration

For an overview of the default configuration files shipped with the platform, see Configuration files.

After completing the installation, you probably need to review and edit most of, if not all, the configuration files so that:

- The specified file and directory paths point to the correct locations and resources on the target system.
- Where applicable, correct user names for the target system are defined as application owners in configuration or initialization files.

The steps may vary, depending on the specified custom installation location for the platform, and on the file structure/organization of the target system.

These are some general guidelines:

- Verify that the `gi-storage` directory exists under `/tmp`. If it does not exist, create it.
- Copy or symlink the provided configuration files to their corresponding custom directories.
- Open the `/etc/eclecticiq/platform_settings.py` platform configuration file, and verify the following parameter values:
  - Environment variable paths should reflect the actual location of the `platform_settings.py` file on the target system.
  - Assign the `/supervisord.d` directory to the user that installed the platform and to the group that user belongs to.
  - Edit the `user` and `stdout_logfile` parameters in all the `.ini` files containing Supervisor configuration for all Supervisor-managed applications. Normally, these files are in the `/supervisord.d` directory:
    - `user`: assign the user that installed the platform.
    - `stdout_logfile`: define a path to a directory to store Supervisor log files.
  - Make sure that Supervisor user has read and write access to `/supervisord.d` directory.
- You may need to run `chmod 755` on the platform root directory to allow file execution. Nginx needs these file permissions to run and to serve the platform UI. To do so, run: `chmod 755 /opt/eclecticiq`.

For further details about the standard configuration and bootstrapping steps following a standard platform installation, see Configure the platform and Bootstrap the platform.

Use these descriptions as guidelines; values such as paths or names for users and groups may vary, depending on the target system hosting the custom platform installation.

Configure and bootstrap

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 installation and configuration guide to
learn what you should do next.
```

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

Install from a tarball archive

As an alternative to installing the platform from a YUM repository and an rpm package, you can donwload a tarball and proceed with the installation from the locally saved tarball archive.

You can install EclecticIQ Platform on a CentOS or RHEL OS in the following ways:

- From the configured YUM repository and an RPM package
- From a downloaded and locally saved tarball archive

To successfully execute several commands in the command line or in the terminal, you may need root-level access rights.
To obtain admin rights, run the following command(s):

```
$ sudo su -
```

Alternatively:

- Grant admin rights to a specific user, who can then log in with their password to perform admin tasks:

```
$ su - ${username}
```

Or:

- Prefix `sudo` to the command you want to run:

```
$ sudo ${command}
```

Install from a tarball archive

If your access permissions to the target system hosting the platform are limited — for example, if you cannot log in as root — you can install the platform to a custom directory.

About the installation from a tarball

Additional dependencies:

- *Python 3.4* needs to be available on the target system.
- Regardless of the specific locations of your Python 3.4 installation, Python `virtualenv` and all standard libraries normally shipped with Python 3.4 should point to the default/typical locations defined during a standard Python installation.

The following dependencies need to be on the target system before you proceed with the platform installation from a tarball:

- *openssl-devel*
- *postgresql-libs*
- *libffi-devel*
- *libldap2-devel*
- *libsasl2-devel*
- *libxml2-devel*
- *libxmlsec1-devel*
- *libxslt-devel*
- *xmlsec1*

Custom installation steps:

- The installation package is delivered as a tarball (*eclecticiq-platform-${version_number}.tar.xz*).
- You need to run a deployment script to begin the installation (*tar-deployer.sh*).
- The currently logged in system user that executes the platform installation is automatically set as the owner of all platform files and directories.
- After completing the installation step, you should check all the configuration files for the platform, its components, and Supervisor-managed applications to make sure that any specified user names, as well as directory or file paths correctly reflect the target system the platform is installed on.
- You may need system administrator rights to run some commands.
  In this case, prefix `sudo` to the command.

Install from a tarball

To install EclecticIQ Platform to a custom location from a source tarball package, do the following:

- Download the *eclecticiq-platform-${version_number}.tar.xz* tarball and the *tar-deployer.sh* deployer script, and save them to a temporary directory.
- Run `chmod 755 tar-deployer.sh` to make the script executable.
- Run `tar-deployer.sh` to install the platform.

```
# Create a directory to install the platform to
$ mkdir eiq-platform

# Copy there the tar archive and the deployer script
$ cp ~/data/eclecticiq-platform-${version_number}.tar.xz ~/data/tar-deployer.sh ./eiq-platform/

# Browse to the platform directory
$ cd eiq-platform/

# Make the script executable
$ chmod 755 tar-deployer.sh

# Run the deployer script
$ ./tar-deployer.sh

# Deployer script output messages
2017-07-13 17:13:12 --- Checking if platform package is available
2017-07-13 17:13:12 --- Extracting package contents
2017-07-13 17:13:19 --- Setting locations
2017-07-13 17:13:20 --- Creating directory structure
2017-07-13 17:13:20 --- Platform deployment complete!
```

Post-installation configuration

For an overview of the default configuration files shipped with the platform, see Configuration files. Remove the first URL element `/opt` from the paths listed on the article to obtain the corresponding relative paths.

After completing the installation, you probably need to review and edit most of, if not all, the configuration files so that:

- The specified file and directory paths point to the correct locations and resources on the target system.
- Where applicable, correct user names for the target system are defined as application owners in configuration or initialization files.

The steps may vary, depending on the specified custom installation location for the platform, and on the file structure/organization of the target system.
These are some general guidelines:

- Verify that the *gi-storage* directory exists under */tmp*.
  If it does not exist, create it.
- Copy or symlink the provided configuration files to their corresponding custom directories.
- Open the */etc/eclecticiq/platform_settings.py* platform configuration file, and verify the following parameter values:
  - Environment variable paths should reflect the actual location of the *platform_settings.py* file on the target system.
  - Assign the */supervisord.d* directory to the user that installed the platform and to the group that user belongs to.
  - Edit the `user` and `stdout_logfile` parameters in all the *.ini* files containing Supervisor configuration for all Supervisor-managed applications.
    Normally, these files are in the */supervisord.d* directory:
    - `user`: assign the user that installed the platform.
    - `stdout_logfile`: define a path to a directory to store Supervisor log files.
      The Supervisor user needs to have read and write access to this directory.

- You may need to run `$ chmod 755` on the platform root directory to allow file execution.
  By default, the platform root directory is */opt/eclecticiq* Nginx needs this user permission configuration to run and to serve the platform UI.

```
$ chmod 755 /opt/eclecticiq
```

For further details about the standard configuration and bootstrapping steps following a standard platform installation, see Configure the platform and Bootstrap the platform.
Use these descriptions as guidelines; values such as paths or names for users and groups may vary, depending on the target system hosting the custom platform installation.

Check permissions

`eclecticiq:eclecticiq` should own Supervisor configuration and log file locations:

```
$ chown -R eclecticiq:eclecticiq /var/run/supervisor/
```

```
$ chown -R eclecticiq:eclecticiq /etc/supervisord.d/
```

```
$ chown -R eclecticiq:eclecticiq /var/log/supervisor/
```

```
$ chown -R eclecticiq:eclecticiq /var/log/eclecticiq/
```

If you reboot the system after applying these changes, ownership of the */var/run/supervisor* directory may be reset to `root:root`.
To address the issue, create a *supervisor.conf* file in the */etc/tmpfiles.d* directory:

```
$ echo 'D /var/run/supervisor 0775 eclecticiq eclecticiq -' > /etc/tmpfiles.d/supervisor.conf
```

Configure and bootstrap

After a successful platform installation, you can proceed to the configuration and bootstrapping steps.

Configure the platform

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

- Required credentials for installation:

```
eiq_user=
eiq_pass=
db_user=test
db_pass=test
neo4j_pass=wXK9pIhDk3
```

- Required variables for installation:

```
server_name=localhost
db_name=platform
data_dir=/media
binary_repo_path=downloads.eclecticiq.com/platform-binary-deps
ssl_certificate=/etc/pki/tls/certs/eiq_platform_local.crt
ssl_certificate_key=/etc/pki/tls/private/eiq_platform_local.key
generate_self_signed_cert=true
```

Credentials and variables listed above need to be exported prior to continuing with the instructions: `export var_name=value` or they can be replaced with their values in any of the commands that contain them.

Configure PostgreSQL

Initialize the database

- Make sure PostgreSQL is not running:

```
# Stop postgres
systemctl stop postgresql-10; sleep 5
```

- Move existing PostgreSqL data directory to new location:

```
# Move postgres data to new location
rsync -av /var/lib/pgsql/ ${data_dir}/pgsql
```

- Remove existing PostgreSqL data directory:

```
# Remove old pgsql data dir
rm -rf /var/lib/pgsql
```

- Create a softlink to the PostgreSQL data directory:

```
# Softlink postgres data dir
ln -s ${data_dir}/pgsql /var/lib/pgsql
```

- Initialize the database:

```
# Initialize the database
/usr/pgsql-10/bin/postgresql-10-setup initdb
```

Enable the database service

- Enable the PostgreSQL service to automatically start at system boot:

```
# Enable the PostgreSQL service
systemctl enable postgresql-10
```

- Restart PostgreSQL by running the following command:

```
# Restart PostgreSQL
systemctl restart postgresql-10
```

- Check if PostgreSQL is running using the following command:

```
# Check if PostgreSQL is running
systemctl status postgresql-10
```

Set the database authentication method

- Modify `/var/lib/pgsql/10/data/pg_hba.conf` configuration file to set the authentication method to `trust` for the `local` type, and to `md5` for the `host` type to request an MD5-encrypted password.

```
# Configure pg_hba.conf
sed -ri 's/^((host|local)\s.*)/#\1/g' /var/lib/pgsql/10/data/pg_hba.conf
echo "local all all peer" >> /var/lib/pgsql/10/data/pg_hba.conf
echo "local all ${db_user} trust" >> /var/lib/pgsql/10/data/pg_hba.conf
echo "host all all samenet md5" >> /var/lib/pgsql/10/data/pg_hba.conf
```

Edit the database configuration

- Depending on your system setup and on the total amount of memory you can allocate to PostgreSQL, you may wish to tune the database configuration. The following extract is just an example for reference, it is not a silver bullet.

```
# Tune PostgreSQL

sed -i '/^\s*shared_buffers\s*=/d' /var/lib/pgsql/10/data/postgresql.conf
echo "shared_buffers = 1024MB" >> /var/lib/pgsql/10/data/postgresql.conf
sed -i '/^\s*work_mem\s*=/d' /var/lib/pgsql/10/data/postgresql.conf
echo "work_mem = 8MB" >> /var/lib/pgsql/10/data/postgresql.conf

# Random page lookups are cheap on SSD disks
sed -i '/^\s*random_page_cost\s*=/d' /var/lib/pgsql/10/data/postgresql.conf
echo "random_page_cost = 1.5" >> /var/lib/pgsql/10/data/postgresql.conf

# Maintenance operations are usually not performed concurrently so they need
# significantly more memory to run faster.
sed -i '/^\s*maintenance_work_mem\s*=/d' /var/lib/pgsql/10/data/postgresql.conf
echo "maintenance_work_mem = 1GB" >> /var/lib/pgsql/10/data/postgresql.conf
```

- Restart the PostgreSQL service.

```
# Restart PostgreSQL
systemctl restart postgresql-10
```

- Check if PostgreSQL is running.

```
# Check PostgreSQL service
systemctl status postgresql-10
```

Create the database

To create a new PostgreSQL database do the following:

- Connect to PostgreSQL with root privileges, access it with the postgres user name, and then create a new database.

```
# Create database
su - postgres -c "psql -c \"CREATE DATABASE ${db_name} WITH encoding = 'UTF-8' lc_ctype =
'en_US.utf8' lc_collate = 'en_US.utf8' template = template0;\""
```

- Create a new user, and assign them a password. Avoid ' symbol in the password.

```
# Create database user
su - postgres -c "psql -c \"CREATE USER ${db_user} WITH PASSWORD '${db_pass}';\""
```

- Grant new user full privileges on the newly created database.

```
# Grant database privileges
su - postgres -c "psql -c \"GRANT ALL PRIVILEGES ON DATABASE ${db_name} TO ${db_user};\""
```

- Verify that user and database are successfully created.

```
# Verify user access
PGPASSWORD=${db_pass} psql -h localhost -U ${db_user} -d ${db_name} -W -c 'select * from
pg_roles' --no-password
```

Update 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.

```
# Add the database to the platform settings
sed -i '/^\s*SQLALCHEMY_DATABASE_URI\s*=/d' /opt/eclecticiq/etc/eclecticiq/platform_settings.py
echo -e "\nSQLALCHEMY_DATABASE_URI =
\"postgresql://${db_pass}:${db_user}@localhost:5432/${db_name}\"" >>
/opt/eclecticiq/etc/eclecticiq/platform_settings.py
```

Configure Nginx

Nginx is the web server we are setting up for the platform. To configure Nginx, do the following:

- Back up nginx.conf:

```
# Backup nginx configuration
conf=/etc/nginx/nginx.conf; backup=$conf.backup.$(date +%s); if [ -f $conf ]; then \mv $conf
$backup; echo "'$conf' backup: '$backup'"; else echo "File '$conf' not found."; fi
```

- Copy to this location the *nginx.conf* file shipped with the platform:

```
# Copy nginx.conf shipped with the platform
\cp /opt/eclecticiq/etc-extras/nginx/nginx.conf /etc/nginx/nginx.conf
```

- Modify nginx configuration:

```
# Configure nginx
sed -ri 's/^\s*user\s.*/user nginx nginx;/g' /etc/nginx/nginx.conf
sed -ri 's/^\s*worker_processes\s.*/worker_processes auto;/g' /etc/nginx/nginx.conf
sed -ri 's/^\s*error_log\s(.*)(\s[a-z]+);/error_log \1 info;/g' /etc/nginx/nginx.conf
sed -ri 's/^(\s*log_format\s+)[a-z]+(\s.*)/\1json\2/g' /etc/nginx/nginx.conf
```

- To enable Nginx to pick up and start the platform configuration, copy the `platform.conf` file shipped with the platform to `/etc/nginx/conf.d`.

```
# Copy platform configuration file for nginx
\cp /opt/eclecticiq/etc/nginx/conf.d/platform.conf /etc/nginx/conf.d/platform.conf
```

- Set server name in platform server configuration for nginx:

```
# Configure server name
sed -ri 's/^(\s*server_name\s+).*;/\1'${server_name};/g' /etc/nginx/conf.d/platform.conf
```

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

```
# Set Nginx certificate paths
sed -ri 's|^\s*(ssl_certificate\s).*|\1'${ssl_certificate};|g' /etc/nginx/conf.d/platform.conf
sed -ri 's|^\s*(ssl_certificate_key\s).*|\1'${ssl_certificate_key};|g'
/etc/nginx/conf.d/platform.conf
```

- To generate a self-signed certificate run the following command(s):

```
# Generate self-signed certificates for nginx
if ${generate_self_signed_cert}; then openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
${ssl_certificate_key} -out ${ssl_certificate} -subj "/C=XX/ST=/L=/O=/CN=test"; fi
```

Do not use self-signed SSL certificates in a production environment. They are meant for development and testing. They are unsuitable for deployment in a live system.

- Enable the Nginx service to automatically start at system boot:

```
# Enable nginx service
systemctl enable nginx
```

- Restart nginx web server:

```
# Restart nginx
systemctl restart nginx
```

- Verify that Nginx is up and running by checking the web server status:

```
# Check if nginx is running
systemctl status nginx
```

Configure Redis

To configure Redis, do the following:

- Back up *redis.conf*:

```
# Backup redis.conf
conf=/etc/redis.conf; backup=$conf.backup.$(date +%s); if [ -f $conf ]; then \mv $conf $backup;
echo "'$conf' backup: '$backup'"; else echo "File '$conf' not found."; fi
```

- Copy to this location the *redis.conf* file shipped with the platform:

```
# Copy redis.conf 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, and the Redis configuration documentation.

- Set redis data directory:

```
# Set redis data directory
sed -ri 's|^(dir\s*).*|\1'${data_dir}'/redis|g' /etc/redis.conf
```

- 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 /media/redis`: sets the location where Redis stores the snapshot database.

Check the service configuration

- Check, and if necessary edit, `/etc/systemd/system/redis.service` to reflect your actual configuration. This is the system configuration file that enables systemd to manage Redis tasks and processes.

Redis will fail to start and run normally if the parameters in this file are not set correctly for the target environment.

Edit the service configuration

- Add a directive to systemd configuration for Redis to give Redis ReadWrite permissions on the data directory:

```
# Make data directory writable to Redis
sed -i -r 's|^(\[Service\])\s*$|\1\nReadWriteDirectories=-'${data_dir}'/redis|g'
/usr/lib/systemd/system/redis.service
```

- Reload systemd configuration daemon:

```
# Reload systemd configuration daemon
systemctl daemon-reload
```

Check permissions

- If the data directory does not exists, create it, and then set `redis` user and `redis` group as the owner:

```
# Set permissions for redis data dir
mkdir -p ${data_dir}/redis; chown -R redis:redis ${data_dir}/redis
```

- `redis:redis` should own also the following locations:

```
# Make redis owner of it's lib and log directories
chown -R redis:redis /var/lib/redis
chown -R redis:redis /var/log/redis
```

- If you have SELinux enabled, add following permissions:

```
semanage fcontext -a -t redis_var_lib_t "${data_dir}/redis(/.*)?" || true
restorecon -iRF /media/redis/ || true
```

Enable the service

- Enable the Redis service to automatically start at system boot:

```
# Enable Redis service
systemctl enable redis
```

- Restart Redis:

```
# Restart Redis
systemctl restart redis
```

- Check if Redis is running:

```
# Check if Redis is running
systemctl status redis
```

- To verify that Redis is working properly:

```
redis-cli -c "ping"
```

Configure Elasticsearch

To configure Elasticsearch, do the following:

- Copy configuration files provided with the platform:

```
# Copy elasticsearch configuration
\cp -R /opt/eclecticiq/etc-extras/elasticsearch/* /etc/elasticsearch/
```

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

```
# Set elasticsearch data dir
sed -i 's|^\s*path\.data:.*|path.data: '${data_dir}'/elasticsearch|g'
/etc/elasticsearch/elasticsearch.yml
```

Check permissions

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

- Create the target directory and make sure the correct user owns it to access it in read and write modes:

```
# Create Elasticsearch data dir and set permissions
mkdir -p ${data_dir}/elasticsearch
chown -R elasticsearch:elasticsearch ${data_dir}/elasticsearch
```

Update platform settings

- Verify that `SEARCH_URL` in `/opt/eclecticiq/etc/eclecticiq/platform_settings.py` points to the correct location corresponding to the Elasticsearch instance:

```
# Configure Elasticsearch url in platform settings
sed -i '/^\s*SEARCH_URL\s*=/d' /opt/eclecticiq/etc/eclecticiq/platform_settings.py
echo "SEARCH_URL = \"http://localhost:9200\"" >>
/opt/eclecticiq/etc/eclecticiq/platform_settings.py
```

Enable the service

- Enable the Elasticsearch service to automatically start at system boot:

```
# Enable elasticsearch service
systemctl enable elasticsearch
```

- Restart Elasticsearch:

```
# Restart Elasticsearch
systemctl restart elasticsearch
```

- Check if Elasticsearch is running:

```
# Verify Elasticsearch is running
systemctl status elasticsearch
```

Configure Logstash

Logstash configuration files are copied automatically to the correct location during the post-install step. You can verify this by listing the configuration directory: `ls -al /opt/eclecticiq/etc/logstash/conf.d/`.

These are the Logstash configuration files related to the platform:

- `conf`
- `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:

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

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

Enable the service

- Enable the Logstash service to automatically start at system boot:

```
# Enable Logstash service
systemctl enable logstash
```

- Restart Logstash:

```
# Restart Logstash
systemctl restart logstash
```

- Check if Logstash is running:

```
# Check if Logstash is running
systemctl status logstash
```

Configure Kibana

Kibana uses an index in Elasticsearch to store saved searches, visualizations, and dashboards.

- Copy Kibana configuration file provided by platform:

```
\cp /opt/eclecticiq/etc-extras/kibana.yml /etc/kibana/
```

Update platform settings

Make sure `/opt/eclecticiq/etc/eclecticiq/platform_settings.py` includes the following parameters:

- `KIBANA_URL`: the IP address of the host machine where the Kibana instance runs, and the port number where it is possible to access Kibana.

```
# update kibana url in platform settings
sed -i '/^\s*KIBANA_URL\s*=/d' /opt/eclecticiq/etc/eclecticiq/platform_settings.py
echo "KIBANA_URL = \"http://localhost:5601\"" >>
/opt/eclecticiq/etc/eclecticiq/platform_settings.py
```

Enable the service

- Enable the Kibana service to automatically start at system boot:

```
# Enable Kibana service
systemctl enable kibana
```

- Restart Kibana:

```
# Restart Kibana
systemctl restart kibana
```

- Check if Kibana is running:

```
# Check if Kibana is running
systemctl status kibana
```

Access Kibana

To access Kibana, in the web browser address bar enter a URL with the following format:

`https://${server_name}/api/kibana/app/kibana#`

Keep the trailing `#`

Configure StatsD

To configure StatsD, do the following:

- Create statsd configuration file at `/opt/statsd/config.js`. Check the following points:
  - Verify that `backends` is assigned `statsd-elasticsearch-backend`, that is, the backend allowing interoperability between Elasticsearch and StatsD.
  - Change the `port` value to the actual port number your Elasticsearch instance uses.
  - Change the `host` value to the actual IP address of the host your Elasticsearch instance runs on.
  - Leave the other parameters and values as is.

```
# Create statsd configuration
echo '
{
  backends:          [ "statsd-elasticsearch-backend" ],
  flushInterval:     30000,  // 30 Seconds
  deleteIdleStats:   true,

  elasticsearch: {
    port:            9200,  // default Elasticsearch port
    host:            "localhost", // Elasticsearch instance IP address
    path:            "/",
    indexPrefix:     "statsd",
    indexTimestamp: "day",
    countType:       "counter",
    timerType:       "timer",
    timerDataType:   "timer_data",
    gaugeDataType:   "gauge",
    formatter:       "default_format"
  }
}' > /opt/statsd/config.js
```

Update platform settings

Make sure `/opt/eclecticiq/etc/eclecticiq/platform_settings.py` includes the following parameters:

- `STATSD_ENABLED`: Boolean switch. Set it to True to enable StatsD.
- `STATSD_HOST`: the IP address of the host machine where the StatsD instance runs. Typically, it corresponds to the IP address of the host machine where the Elasticsearch instance runs.
- `STATSD_PORT`: the port number for the statsd server to communicate. Default value: 8125

```
# Update statsd options in platform settings
sed -i '/^\s*STATSD_ENABLED\s*=/d' /opt/eclecticiq/etc/eclecticiq/platform_settings.py
echo "STATSD_ENABLED = True" >> /opt/eclecticiq/etc/eclecticiq/platform_settings.py
sed -i '/^\s*STATSD_HOST\s*=/d' /opt/eclecticiq/etc/eclecticiq/platform_settings.py
echo "STATSD_HOST = \"localhost\"" >> /opt/eclecticiq/etc/eclecticiq/platform_settings.py
sed -i '/^\s*STATSD_PORT\s*=/d' /opt/eclecticiq/etc/eclecticiq/platform_settings.py
echo "STATSD_PORT = 8125" >> /opt/eclecticiq/etc/eclecticiq/platform_settings.py
```

Map StatsD to Elasticsearch

Do not enable or start the service yet.

For Elasticsearch to correctly load and index StatsD-specific fields, and for Kibana to display StatsD data, you need to locally save and upload to Elasticsearch the StatsD mapping template.

- The platform ships with a mapping template that should not require any customization, so that you can use it as is: `/opt/eclecticiq/etc-extras/statsd/statsd_elasticsearch_template.json`.
- Upload *statsd_elasticsearch_template.json* to the Elasticsearch instance, to the `/_template/statsd-template` directory:

```
# Push statsd mapping template to Elasticsearch (may have to run this several times if not successful the first time)
curl -X POST http://127.0.0.1:9200/_template/statsd-template -d@/opt/eclecticiq/etc-extras/statsd/statsd_elasticsearch_template.json
```

Restart Elasticsearch

```
# Restart elasticsearch
systemctl restart elasticsearch; sleep 5
```

- When Elasticsearch has restarted, delete any existing indices so that statsd creates new ones with correct mapping:

```
# Delete existing statsd indices
curl -XDELETE 'http://localhost:9200/statsd-*'
```

This creates a new index in Elasticsearch: `statsd-*`. StatsD-specific index fields are:

- `ns`: namespace - it refers to a platform component, such as API, ingestion, tasks, and so on.
- `grp`: group - it refers to an event related to the platform area specified in ns.
- `tgt`: target - it reports a specific platform artifact, such as entities and feeds.
- `act`: action - it reports the type of action the entry record logged.

statsd-elasticsearch-backend

statsd-elasticsearch-backend implements a backend for Elasticsearch to work with StatsD.

Install statsd-elasticsearch-backend as a Node.js module:

- Download and install statsd-elasticsearch-backend:

```
# Install statsd-elasticsearch-backend
escaped_username=$(echo $eiq_user | sed 's|\@|%40|g')
( cd /opt/statsd; npm install https://${escaped_username}:${eiq_pass}@${binary_repo_path}/statsd-elasticsearch-backend-0.4.2.tar.gz )
```

Enable the service

- Create systemd entry for statsd service:

```
# create statsd service
echo "
[Unit]
Description=network daemon to collect metrics

[Service]
User=eclecticiq
Type=simple
ExecStart=/opt/statsd/bin/statsd /opt/statsd/config.js
Restart=on-failure

[Install]
WantedBy=multi-user.target
" > /etc/systemd/system/statsd.service
```

- Enable the StatsD service to automatically start at system boot:

```
# Enable StatsD service
systemctl enable statsd
```

- Restart StatsD:

```
# Restart StatsD
systemctl restart statsd
```

- Check if StatsD is running:

```
#Check if StatsD is running
systemctl status statsd
```

Configure Neo4j

- Copy Neo4j configuration provided by platform:

```
# Copy Neo4j config
\cp /opt/eclecticiq/etc-extras/neo4j/neo4j.conf /etc/neo4j/
```

- Set database location to your data directory:

```
# Set Neo4j database location
mkdir -p ${data_dir}/neo4j/
chown -R neo4j:neo4j ${data_dir}/neo4j
sed -ri 's|^(\s*dbms.directories.data\s*=).*|\1'${data_dir}/neo4j'|g' /etc/neo4j/neo4j.conf
```

Disable remote shell

- Disable Neo4j remote shell for security reasons:

```
# Disable Neo4j remote shell
sed -i '/^\s*dbms.shell.enabled\s*=/d' /etc/neo4j/neo4j.conf
echo "dbms.shell.enabled=false" >> /etc/neo4j/neo4j.conf
```

Enable the service

- Enable the Neo4j service to automatically start at system boot:

```
# Enable Neo4j service
systemctl enable neo4j
```

- Restart Neo4j:

```
# Restart Neo4j
systemctl restart neo4j
```

- Check if Neo4j is running:

```
# Check if Neo4j is running
systemctl status neo4j
```

Configure authentication

- Set Neo4j password:

```
# Set neo4j password
curl -H "Content-Type: application/json" -X POST -d '{"password":"'${neo4j_pass}'"}' -u
neo4j:neo4j http://localhost:7474/user/neo4j/password
```

- Update platform configuration:

```
# Configure neo4j credentials in platform settings
sed -ri 's/^\s*(NEO4J_USER\s*=\s*).*/\1'\''neo4j\''/g'
/opt/eclecticiq/etc/eclecticiq/platform_settings.py
sed -ri 's/^\s*(NEO4J_PASS\s*=\s*).*/\1'\''${neo4j_pass}\''/g'
/opt/eclecticiq/etc/eclecticiq/platform_settings.py
```

Configure Postfix

Enable the service

- Enable the Postfix service to automatically start at system boot:

```
# Enable the Postfix service
systemctl enable postfix
```

- Restart Postfix:

```
# Restart Postfix
systemctl restart postfix
```

- Check if Postfix is running:

```
# Check if Postfix is running
systemctl status postfix
```

Update the platform settings

Set secret key and session token

When you install, update or reinstall the platform, change the following default values in the `/opt/eclecticiq/etc/eclecticiq/platform_settings.py` configuration file:

- `SECRET_KEY`: change the default value with a randomly generated one.
  - You can use any cryptographic pseudorandom value generator. Recommended: `/dev/urandom`
  - The generated secret key needs to be at least 32 character long.
  - If the secret key value contains single quotes `'`, escape them by prepending a backslash. Example: `\'`
  - If you leave the secret key value field empty, or if the secret key value is shorter than 32 characters, the platform won't start.

```
# Generate platform secret key
secret=$(</dev/urandom tr -dc a-zA-Z0-9 | fold -w 32 | head -n 1)
sed -i '/^\s*SECRET_KEY\s*=/d' /opt/eclecticiq/etc/eclecticiq/platform_settings.py
echo "SECRET_KEY = '$secret'" >> /opt/eclecticiq/etc/eclecticiq/platform_settings.py
```

- `JWT_EXPIRATION_DELTA = 60 * 30`: change the expiration time of the user authentication token as needed. The value is measured in seconds.
  - By default, the token expires 30 minutes 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.

```
# Set login expiration time
sed -i '/^\s*JWT_EXPIRATION_DELTA\s*=/d' /opt/eclecticiq/etc/eclecticiq/platform_settings.py
echo "JWT_EXPIRATION_DELTA = 60 * 30" >> /opt/eclecticiq/etc/eclecticiq/platform_settings.py
```

Configure SSL/HTTPS in Nginx/Neo4j

Configure SSL certificates in Nginx, Enable client certificate verification, and set up HTTPS and user authentication in Neo4j.

Configure SSL certificates in Nginx

- To enable Nginx to pick up and start the platform configuration, copy the *platform.conf* file shipped with the platform 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:

```
$ vi /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** `(https://nginx.org/en/docs/http/configuring_https_servers.html)` point to the correct locations where your certificates are stored.

  Example:

```
ssl_certificate      /etc/pki/tls/certs/${certificate_name}.crt;
ssl_certificate_key  /etc/pki/tls/private/${key_name}.key;
```

To generate a self-signed certificate run the following command(s):

```
$ openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/pki/tls/private/platform_local.key -out /etc/pki/tls/certs/platform_local.crt
```

Warning:
Do not use self-signed SSL certificates in a production environment. They are meant for development and testing. They are unsuitable for deployment in a live system.

Enable client certificate verification

Nginx supports client certificate verification through the following directives:

- `ssl_client_certificate`
- `ssl_verify_client`

Example:

```
server {
  listen      10.0.1.128:443;
  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 the certificate used to sign the client certificates. You can obtain this file from a certification authority (CA).

Install a client certificate in Google Chrome

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

- Go to **Settings > Advanced > Privacy and security > Manage certificates** .
- On the **Your certificates** tab, click **Import** to import your client certificate.

To set up a certification authority (CA) and to generate client and server certificates, OpenSSL is the recommended tool.

Enable the service

- Enable the Nginx service to automatically start at system boot:

```
$ systemctl enable nginx
```

- Start the Nginx web server:

```
$ systemctl start nginx
```

- Verify that Nginx is up and running by checking the web server status:

```
$ systemctl status nginx
```

Configure HTTPS and authentication in Neo4j

If Neo4j is hosted on a server that has network access, it is a good idea to enable secure access and user access control through HTTPS/SSL and an authentication mechanism.

Enable secure access

To enable access to Neo4j through a secure connection, you need to:

- Set the transport protocol to HTTPS.
- Set a secure port.
- Provide a valid SSL key and certificate pair.
  On first-time start, Neo4j automatically generates a self-signed SSL certificate and a private key.
  For production environments, replace these files with your own SSL key and certificate.

By default, neo4j.conf port for HTTPS access is `7473`. Set it to a different port.

The **Neo4j official documentation** `(https://neo4j.com/docs/operations-manual/3.3/)` provides comprehensive instructions and reference on enabling SSL and HTTPS, as well as setting certificates and keys.

These tasks require editing the */etc/neo4j/neo4j.conf* and the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* configuration files.

- *neo4j.conf*: **define a policy with SSL certificate and key** `(https://neo4j.com/docs/operations-manual/current/security/ssl-framework/#_policy_definition).`
- *platform_settings.py*: add or edit the following parameters, so that the platform is aware that communication with Neo4j flows through HTTP and SSL.

```
# All values are dummy: replace them.

# Specify HTTPS and secure port for Neo4j
NEO4J_URL = "http://127.0.0.1:7473"

# Specify SSL cert for Neo4j
NEO4J_SSL_VERIFY = "/etc/neo4j/ssl/snakeoil.cert"
```

- The port number you assign to `NEO4J_URL` should be the same as the port number you set in the `dbms.connector.https` **connector** `(https://neo4j.com/docs/operations-manual/3.3/configuration/connectors/)` in the *neo4j.conf* configuration file.

- The value you assign to `NEO4J_SSL_VERIFY` should point to the SSL certificate file whose path you set in the `dbms.ssl.policy` parameter group, where:
  - `dbms.ssl.policy.${policy-name}.base_directory=etc/neo4j/ssl`
  - `dbms.ssl.policy.${policy-name}.public_certificate=snakeoil.cert`

Enable authentication

To enable authentication for Neo4j, you need to:

- Enable authentication in Neo4j.
- Set the correct Neo4j login user name and password in the platform settings file.

These tasks require editing the */etc/neo4j/neo4j.conf* and the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* configuration files.

- *neo4j.conf*: **enable authentication and authorization** (https://neo4j.com/docs/operations-manual/current/security/authentication-authorization/enable/).
  Optionally, you can **set an initial password** (https://neo4j.com/docs/operations-manual/current/configuration/set-initial-password/).
- *platform_settings.py*: add or edit the following parameters, so that the platform is aware that it needs to log in to Neo4j with the credentials specified in *neo4j.conf*:

```
# Default Neo4j user name to log in to Neo4j
NEO4J_USER = "neo4j"

# Default Neo4j password to log in to Neo4j
NEO4J_PASSWORD = "neo4j"
```

Change any default values for user name and password as appropriate.
If you do not need to enable authentication, remove the `NEO4J_USER` parameter, or set it to `NEO4J_USER = ""`.

Further reference about Neo4j set up and configuration:

- **Neo4j default file locations** (https://neo4j.com/docs/operations-manual/3.3/configuration/file-locations/)
- **Neo4j ports** (https://neo4j.com/docs/operations-manual/3.3/configuration/ports/)
- **Neo4j configuration settings** (https://neo4j.com/docs/operations-manual/3.3/reference/configuration-settings/)
- **Neo4j status codes** (https://neo4j.com/docs/developer-manual/3.3/reference/status-codes/)
- **Security checklist** (https://neo4j.com/docs/operations-manual/3.3/security/checklist/)

Enable the service

- Enable the Neo4j service to automatically start at system boot:

```
systemctl enable neo4j
```

- To start Neo4j, run the following command(s):

```
$ systemctl start neo4j
```

- To check if Neo4j is running, run the following command(s):

```
$ systemctl status neo4j
```

Update the platform settings

Update the platform settings so that the configured values correctly reflect the platform host environment, and set up LDAP, LDAP with Active Directory (AD), or SAML authentication mechanisms.

After setting up dependencies and third-party components, you can proceed to update the platform settings.

The platform stores its core configuration settings in the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* file. The main sections in this file are:

- Platform configuration, including settings to interoperate with dependencies and third-party products
- LDAP authentication configuration
- SAML authentication configuration.

Update platform_settings.py

- 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:

```
"""
Default settings module.

These are just some application defaults and will be overridden by
a custom settings file.
"""

# Make sure any settings used by the applications are listed here;
# that makes it easier to see which settings are available.

# URLs to access the API
APPLICATION_PREFIX = '/private'
APPLICATION_PREFIX_PUBLIC = '/api'

# Logging
LOG_LEVEL = 'warning,eiq:info'

# Error handling
GENERIC_ERROR_HANDLING = True

# Database
SQLALCHEMY_TRACK_MODIFICATIONS = True
ELASTICSEARCH_QUERY_TIMEOUT = '20s'

# Security
COMPONENT_SHARED_SECRET = '${component_secret_key_value}'
SECRET_KEY = '${secret_key_value}'
JWT_AUTH_ENDPOINT = 'auth'
JWT_EXPIRATION_DELTA = 60 * 30  # 30 minutes

# Optionally specify custom a CA bundle for outgoing requests
REQUESTS_CA_BUNDLE = None
```

```
# If an installed extension should not be loaded, its name should be
# added to this list
DISABLED_EXTENSIONS = []

# CORE EXTENSION SETTINGS
# Feeds with mount point transport type can access only the dir paths in these lists.
# Example: ["/mnt/", "/media/"]
# Incoming feeds, allowed mount point dirs (and their subdirs)
MOUNT_POINT_POLL_ALLOWED_DIRECTORIES = []
# Outgoing feeds, allowed mount point dirs (and their subdirs)
MOUNT_POINT_PUSH_ALLOWED_DIRECTORIES = []

# Default half-life values in days
HALF_LIFE = {
    'campaign': 182,
    'course-of-action': 182,
    'exploit-target': 182,
    'incident': 182,
    'indicator': 182,
    'ttp': 182,
    'threat-actor': 182,
    'report': 182
}

# External components and services
SQLALCHEMY_DATABASE_URI = 'postgresql://${username}:${password}@localhost:5432/platform'
SQLALCHEMY_POOL_SIZE = 20
SQLALCHEMY_MAX_OVERFLOW = 20
# Neo4j 7473 means that HTTPS is enabled and Neo4j SSL certificates are set
NEO4J_URL = 'https://127.0.0.1:7473'
NEO4J_SSL_VERIFY = '/etc/neo4j/ssl/snakeoil.cert'
NEO4J_USER = 'neo4j'
NEO4J_PASSWORD = 'changeme'
NEO4J_BATCHING_URL = 'http://127.0.0.1:4008'
NEO4J_DEBUG = False
REDIS_URL = 'redis://127.0.0.1:6379/'
SEARCH_URL = 'http://127.0.0.1:9200'
KIBANA_URL = 'http://127.0.0.1:5601'
STATSD_ENABLED = True
STATSD_HOST = '127.0.0.1'
STATSD_PORT = 8125
ENRICHMENT_URL = ''

SUPERVISORD_HOSTS = '127.0.0.1:9001'  # Comma separated list of host:port

SYSTEMD_SERVICES = [
    'elasticsearch',
    'logstash',
    'postfix',
    'neo4j',
    'postgresql-10',
    'redis',
    'statsd',
    'kibana',
]

DISCOVERY_WORKSPACE_SIZE_LIMIT = 500
DISCOVERY_SEARCH_LIMIT = 1000
DISCOVERY_RESULTS_LIMIT = 500

GRAPH_INGESTION_SPOOL_DIRECTORY = '/opt/eclecticiq/tmp/gi-storage'
GRAPH_QUEUE_NAME = 'queue:graph:inbound'
SEARCH_QUEUE_NAME = 'queue:search:inbound'

PROXY_URL_FILE_PATH = '/opt/eclecticiq/etc/eclecticiq/proxy_url'
PLATFORM_MANIFESTS_DIR = '/opt/eclecticiq/manifests'
```

```
# Ingestion profiling output: use a deterministic directory as a default
# to make it easier for others to find it (and zip it and send it).
PROFILING_OUTPUT_DIRECTORY = os.path.join(
    tempfile.gettempdir(), 'eiq-platform-profiling')

PROFILING_ON_START = False

# Enricher settings
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/{}'

# Email server settings
# Check host settings in etc/postfix/main.cf
# if Postfix is on the same host as the platform
SMTP_HOST = 'localhost'
SMTP_PORT = 25
SMTP_USERNAME = None
SMTP_PASSWORD = None
# If you enable TLS or SSL, make sure the settings here
# are reflected in the Postfix configuration, too: etc/postfix/main.cf
# Allowed values: None, 'tls', 'ssl'
SMTP_ENCRYPTION = None

# Celery settings
BROKER_URL = REDIS_URL
CELERY_RESULT_BACKEND = REDIS_URL
CELERY_ENABLE_UTC = True
CELERY_TASK_SERIALIZER = 'json'
CELERY_RESULT_SERIALIZER = 'json'
CELERY_ACCEPT_CONTENT = ['json']
CELERYD_LOG_FORMAT = '%(message)s'
CELERYD_TASK_LOG_FORMAT = '%(message)s'
CELERYD_HIJACK_ROOT_LOGGER = False
CELERY_REDIRECT_STDOUTS_LEVEL = 'DEBUG'
CELERY_TRACK_STARTED = True
CELERY_TASK_RESULT_EXPIRES = 1 * 60 * 60    # 1 hour
CELERY_STORE_ERRORS_EVEN_IF_IGNORED = True

CELERYBEAT_SCHEDULER = (
    'eiq.platform.taskrunner.scheduler.DynamicScheduler')

CELERY_QUEUES = [
    {'name': 'enrichers',
     'routing_key': 'eiq.enrichers.#'},

    {'name': 'enrichers-priority',
     'routing_key': 'priority.enrichers.#'},

    {'name': 'incoming-transports',
     'routing_key': 'eiq.incoming-transports.#'},

    {'name': 'incoming-transports-priority',
     'routing_key': 'priority.incoming-transports.#'},

    {'name': 'outgoing-transports',
     'routing_key': 'eiq.outgoing-transports.#'},

    {'name': 'outgoing-transports-priority',
     'routing_key': 'priority.outgoing-transports.#'},

    {'name': 'outgoing-feeds',
     'routing_key': 'eiq.outgoing-feeds.#'},
```

```
    {'name': 'outgoing-feeds-priority',
     'routing_key': 'priority.outgoing-feeds.#'},

    {'name': 'utilities',
     'routing_key': 'eiq.utilities.#'},

    {'name': 'utilities-priority',
     'routing_key': 'priority.utilities.#'},

    {'name': 'discovery',
     'routing_key': 'eiq.discovery.#'},

    {'name': 'discovery-priority',
     'routing_key': 'priority.discovery.#'},

    {'name': 'entity-rules-priority',
     'routing_key': 'priority.entity-rules.#'},

    {'name': 'extract-rules-priority',
     'routing_key': 'priority.extract-rules.#'},

    {'name': 'reindexing',
     'routing_key': 'eiq.reindexing.#'},
]

# Time limits for Celery tasks

# Generic limits per task families in seconds
CELERY_TIME_LIMIT_PER_FAMILY = {
    'eiq.enrichers': 2 * 60,  # 2 min
    'eiq.incoming-transports': 8 * 60 * 60,  # 8 hours
    'eiq.outgoing-transports': 2 * 60,  # 2 min
}

# Limits for specific tasks in seconds
CELERY_TIME_LIMIT_PER_TASK = {
    'eiq.utilities.update_taxonomy_in_search': 60,
    'eiq.utilities.delete_taxonomy_in_search': 60,
    'eiq.utilities.send_email': 30,
    'eiq.utilities.taxii_discovery': 60,
    'eiq.utilities.postponed_entity_signals': 60,
    'eiq.utilities.create_notifications': 60,

    'eiq.discovery.search_discovery': 10 * 60,  # 10 min
    'eiq.discovery.delete_discovery': 60,

    'eiq.entity-rules.entity_rule_task': 2 * 60 * 60,  # 2hours,
    'eiq.extract-rules.extract_rule_task': 2 * 60 * 60,  # 2hours,
}
```

Warning:
Set secret key and session token

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

- `SECRET_KEY = ''`: change the default empty value with a randomly generated one.
  - You can use any cryptographic pseudorandom value generator.
    Recommended: */dev/urandom*
    Example: `cat /dev/urandom | tr -dc a-zA-Z0-9[:punct:] | fold -w 32 | head -n 1`
  - The generated secret key needs to be at least 32 character long.
  - If the secret key value contains single quotes (`'`), escape them by prepending a backslash. Example: `\'`
  - If you leave the secret key value field empty, or if the secret key value is shorter than 32 characters, the platform won't start.
- `JWT_EXPIRATION_DELTA = 60 * 30`: change the expiration time of the user authentication token as needed. The value is measured in seconds.

By default, the token expires 30 minutes after successfully signing in to a platform user session. When the token expires, 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 60 seconds. 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 - 1 minute*.

Configure LDAP authentication

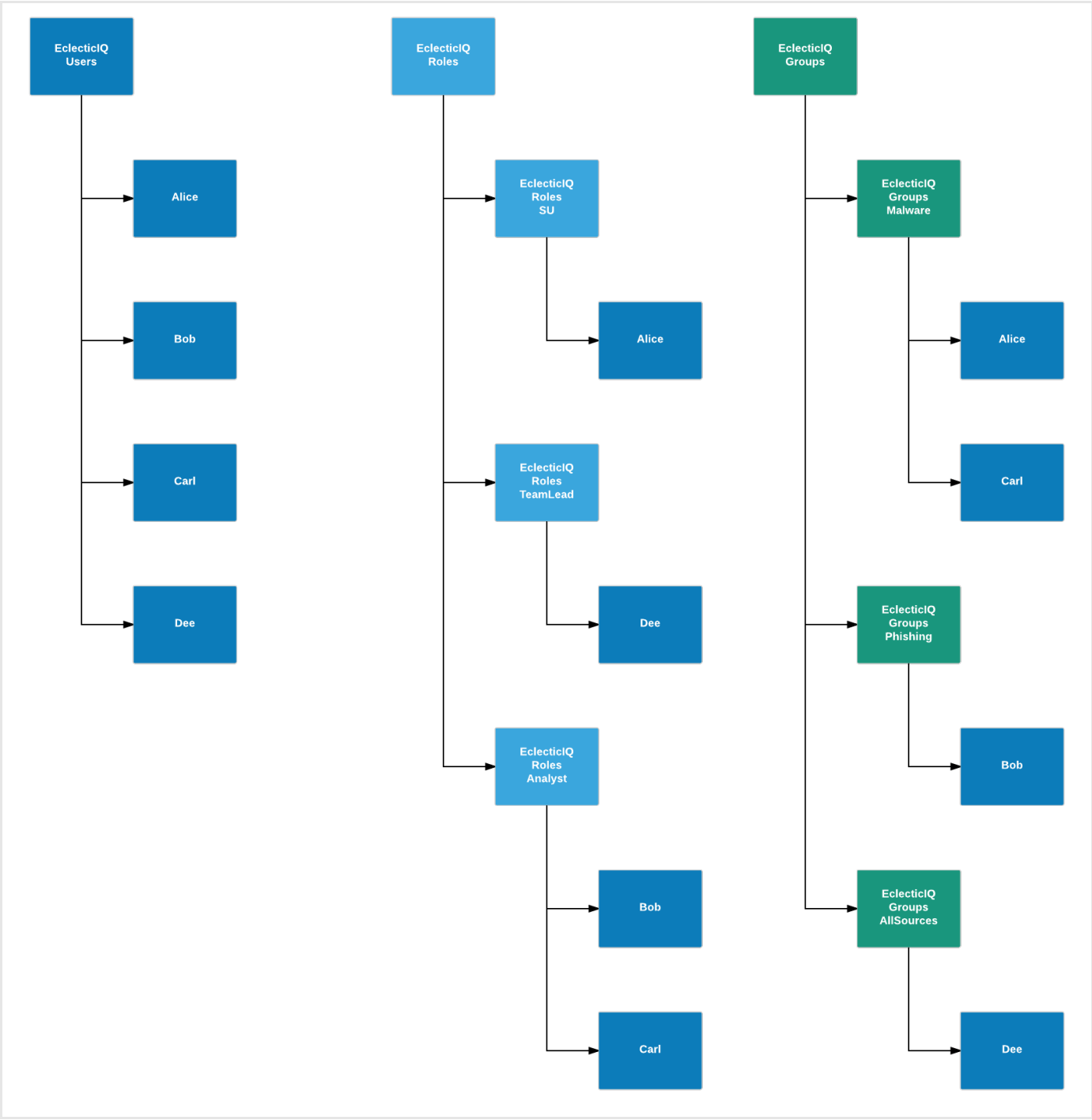
You can configure the platform to import users and groups from an LDAP directory service. It is preferable not to mix LDAP users and local users in the platform. If you create local users in the platform, and then import LDAP users, LDAP users override the local ones.

Therefore, if you implement authentication through LDAP, it is better to delegate user access right management entirely to the LDAP service.

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:
    - `EclecticIQUsers`
    - `EclecticIQRoles`
    - `EclecticIQGroups`
- Create LDAP *users* as necessary.
  - Assign these users to the `EclecticIQUsers` 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.
- 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.
- 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 role and group name 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`: specifiy the valid credentials to bind to the LDAP connection, search for users, read groups and roles.
- `LDAP_BIND_PASSWORD`: specify the password associated with the bnding 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=EclecticIQUsers,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** `(http://ldapwiki.com/wiki/best%20practices%20for%20ldap%20naming%20attributes)` used to extract a user's first/given name.

- `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.
  Alternatively, you can grant a user admin rights by assigning them to the admin group whose name is specified in `LDAP_ADMIN_ROLE_GROUP_NAME`.

  To grant a user admin rights, assign `LDAP_USER_IS_ADMIN_ATTR` the following value: `"isEclecticIQAdmin"`.

  Example:

```
LDAP_USER_IS_ADMIN_ATTR = "isEclecticIQAdmin"
```

- `LDAP_ADMIN_ROLE_GROUP_NAME`: to grant a user admin rights, assign them to the admin group whose name is specified in `LDAP_ADMIN_ROLE_GROUP_NAME`.
  For example, if you assign the attribute the `"EclecticIQAdminsGroup"` value, you can grant a user admin rights by assigning them to the *EclecticIQAdminsGroup* group.

  Alternatively, you can grant a user admin rights by assigning them the `"isEclecticIQAdmin"` value defined in `LDAP_USER_IS_ADMIN_ATTR`.

  Example:

```
LDAP_ADMIN_ROLE_GROUP_NAME = "EclecticIQAdminsGroup"
```

Example LDAP configuration:

```
# LDAP settings

LDAP_AUTH_ENABLED = True
LDAP_URI = 'ldap://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=EclecticIQUsers,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_CASE_SENSITIVE_MATCHING = True

LDAP_USER_IS_ADMIN_ATTR = 'isEclecticIQAdmin'
LDAP_ADMIN_ROLE_GROUP_NAME = 'EclecticIQAdminsGroup'
```

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).

- As with LDAP, ensure you represent platform roles and groups as AD groups.
- AD 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 role and group name values in the platform.

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 an LDAP setup that enables interoperability with AD:

LDAP	AD
-	`memberOf:1.2.840.113556.1.4.1941`
`objectClass=posixGroup`	`objectClass=group`

LDAP	AD
`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)` 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 successfully execute several commands in the command line or in the terminal, you may need root-level access rights.
To obtain admin rights, run the following command(s):

```
$ sudo su -
```

Alternatively:

- Grant admin rights to a specific user, who can then log in with their password to perform admin tasks:

```
$ su - ${username}
```

Or:

- Prefix `sudo` to the command you want to run:

```
$ sudo ${command}
```

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

```
$ 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'
```

Configure SAML authentication

You can configure the platform to authenticate users with **SAML** (https://www.oasis-open.org/standards#samlv2.0). **SAML** (https://docs.oasis-open.org/security/saml/v2.0/) is an XML-based format to exchange authentication and authorization data, usually between an identity provider and a service provider.

It is preferable not to mix SAML users and local users in the platform, as SAML-authenticated users can override local platform users. When such an override occurs, the platform recognizes SAML users as internal, that is, local, and local platform users as external.

The current SAML implementation in the platform supports SAML authentication and authorization, but not SSO.

This is the **SAML flow implemented in the platform** (https://en.wikipedia.org/wiki/saml_2.0#sp_redirect_request.3b_idp_post_response):

- The user makes a SAML sign in request to the platform.
- The platform acts as the service provider, and it redirects the user to the identity provider.
- The identity provider identifies the user, and it issues a document back to the user.
- The user makes a request to the service provider to pass a token, and then the user requests the target resource (for example, a web page).
- The service provider delivers the requested resource.

To implement SAML authentication, you need to:

- Set up and configure a SAML identity provider.
- Configure the platform to perform authentication through SAML as a service provider.
- Generate a metadata XML file, so that the identity provider can recognize the platform instance as a legitimate service provider.

To configure SAML authentication in the platform, 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.
- SAML role and group names, that is, the values of the `SAML_USER_ROLES_ATTR` and `SAML_USER_GROUPS_ATTR` attributes, should exactly match the corresponding role and group name values in the platform.
- `SAML_AUTH_ENABLED`: Boolean switch to enable/disable authentication.
  Default value: `False` (SAML disabled).
  To enable SAML authentication, set it to `True`.
- `SAML_IDP_METADATA`: it can contain *either* a `url` *or* a `file` parameter (not both):
  - `url`: a valid URL pointing to the XML metadata the identity provider needs to access to recognize the platform instance as a legitimate service provider.
  - `file`: a valid path pointing to the XML metadata file the identity provider needs to access to recognize the platform instance as a legitimate service provider.
- `SAML_IDP_ENTITYID`: define a URI that uniquely identifies the configured SAML identity provider.
  The URI should not exceed 1024 characters. The SAML 2.0 core specification recommends that a system entity use a URL containing its own domain name to identify itself.
- `SAML_USER_USERID_ATTR`: define the naming attribute used to extract a user ID.
- `SAML_USER_EMAIL_ATTR`: define the naming attribute used to extract a user's email address.
- `SAML_USER_GROUPS_ATTR`: define the naming attribute used to extract a user's group details.

- `SAML_USER_ROLES_ATTR`: define the naming attribute used to extract a user's role details.
- `SAML_USER_FIRSTNAME_ATTR`: define the naming attribute used to extract a user's first/given name.
- `SAML_USER_LASTNAME_ATTR`: define the naming attribute used to extract a user's surname/family name.
- `SAML_USER_IS_ADMIN_ATTR`: specify if the user should be granted admin rights.
  To grant a user admin rights, assign the attribute the following value: `"isEclecticIQAdmin"`.
  Example:

```
SAML_USER_IS_ADMIN_ATTR = "isEclecticIQAdmin"
```

- `SAML_ADMIN_ROLE_GROUP_NAME`: if you grant a user admin rights with `SAML_USER_IS_ADMIN_ATTR`, you also need to add them to a group that has admin rights and permissions.
  To add a user to an admin group, assign the attribute the following value: "EclecticIQAdminsGroup".
  Example:

```
SAML_ADMIN_ROLE_GROUP_NAME = "EclecticIQAdminsGroup"
```

- `SAML_SIGN_AUTHN_REQ`: Boolean switch to enable/disable manadatory digital signing for authentication requests.
  Default value: `False` (authentication request signing disabled)
- `SAML_WANT_ASSERT_SIGNED`: Boolean switch to enable/disable manadatory digital signing for assertions received from the identity provider.
  Default value: `True` (assertions must be signed)
- `SAML_WANT_RESPONSE_SIGNED`: Boolean switch to enable/disable manadatory digital signing for responses received from the identity provider.
  Default value: `False` (response signing disabled)
- `SAML_ENC_KEY`: define the path to the location where the decryption key is stored.
  The key should include a `Recipient` attribute defining the entity the key was encrypted for.
  The value of the `Recipient` attribute should be the URI identifier of a SAML system entity.
- `SAML_ENC_CERT`: define the path to the location where the decryption certificate is stored.
- `SAML_XMLSEC_BIN`: define the path to the local ***xmlsec*** `(https://pypi.python.org/pypi/xmlsec)` installation.
  xmlsec provides Python bindings for the XML Security Library.
- `SAML_METADATA_ORG`: define metadata information to identify the platform instance or the organization.
  - `name`: define one or more names to identify the platform instance or the organization, along with the corresponding language identifier(s).
    Names you define here may or may not be suitable for human consumption.
  - `display_name`: define one or more names to identify the platform instance or the organization, along with the corresponding language identifier(s).
    Names you define here are suitable for human consumption.
  - `url`: define one or more URIs pointing to resources users can look up for additional information.
    For example, a corporate web site, a blog, a publicly accessible wiki page, and so on.

- `SAML_METADATA_CONTACT_PERSON`: define metadata information to identify a designated contact person for the platform instance or the organization.
  - `given_name`: define the contact person's first/given name.
  - `sur_name`: define the contact person's surname/family name.
  - `email_address`: define the contact person's email address.
  - `contact_type`: define the contact person's professional role.
    For example, **system administrator** `(https://xkcd.com/705/)`, **security officer** `(https://xkcd.com/327/)`, **software engineer** `(https://xkcd.com/378/)`, and so on.

Example

```
SAML_AUTH_ENABLED = False

SAML_IDP_METADATA = {
    'url': 'https://idp.testshib.org/idp/shibboleth',
    # 'file': '/idp-metadata.xml'
}

SAML_IDP_ENTITYID = "https://idp.testshib.org/idp/shibboleth"

# Required attributes
SAML_USER_USERID_ATTR = 'uid'
SAML_USER_EMAIL_ATTR = 'email'
SAML_USER_GROUPS_ATTR = 'EclecticIQGroups'
SAML_USER_ROLES_ATTR = 'EclecticIQRoles'

# Optional attributes
SAML_USER_FIRSTNAME_ATTR = 'givenName'
SAML_USER_LASTNAME_ATTR = 'sn'

SAML_USER_IS_ADMIN_ATTR = 'isEclecticIQAdmin'
SAML_ADMIN_ROLE_GROUP_NAME = 'EclecticIQAdminsGroup'

SAML_SIGN_AUTHN_REQ = False
SAML_WANT_ASSERT_SIGNED = True
SAML_WANT_RESPONSE_SIGNED = False

SAML_ENC_KEY = '/tmp/saml-keys/platform-saml-key.pem'
SAML_ENC_CERT = '/tmp/saml-certs/platform-saml-cert.crt'

SAML_XMLSEC_BIN = '/usr/bin/xmlsec1'

SAML_CASE_SENSITIVE_MATCHING = True

SAML_METADATA_ORG = {
    'name': 'Gus Meth Lab',
    'display_name': [('Los Pollos Hermanos', 'es-mx')],
    'url': 'http://lospolloshermanos.com',
}

SAML_METADATA_CONTACT_PERSON = {
    'given_name': 'Gus',
    'sur_name': 'Fring',
    'email_address': ['gus.fring@lospolloshermanos.com'],
    'contact_type': 'General manager',
}
```

Test SAML authentication

To test SAML authentication in the platform, you can use the following example as a guideline.

To successfully execute several commands in the command line or in the terminal, you may need root-level access rights.
To obtain admin rights, run the following command(s):

```
$ sudo su -
```

Alternatively:

- Grant admin rights to a specific user, who can then log in with their password to perform admin tasks:

```
$ su - ${username}
```

Or:

- Prefix `sudo` to the command you want to run:

```
$ sudo ${command}
```

- Open the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* platform settings file, and enable SAML authentication.
- Generate an SSL key/certificate pair.
  You can use **this handy Bash script** `(https://gist.github.com/adrianwebb/3313395)` to do that (requires **OpenSSL** `(https://www.openssl.org/)`).
- Save the generated files to a directory.
- Specify the correct paths to the key and the certificate in the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* platform settings file:

```
SAML_AUTH_ENABLED = True
SAML_ENC_KEY = '/tmp/saml-keys/platform-saml-key.pem'
SAML_ENC_CERT = '/tmp/saml-certs/platform-saml-cert.crt'
```

- Restart the `platform-api` service:

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

- Go to the platform sign in page, and verify that the page includes the option to sign in with SAML.

- Generate the SAML service provider metadata, which you need to register the platform instance with the designated identity provider:
  - Log in to the platform host system.
  - Change user to `eclecticiq`.
  - As `eclecticiq` user, run the following command(s):

```
$ source /opt/eclecticiq/platform/api/bin/activate
(api) $ export EIQ_PLATFORM_SETTINGS="/opt/eclecticiq/etc/eclecticiq/platform_settings.py"
(api) $ eiq-platform saml generate-metadata --output-file /tmp/saml-serviceprovider-metadata.xml
```

- Go to the save-to directory — */tmp* — in the example and verify that the XML metadata file exists.
- Upload, send or transmit the metadata file to the configured SAML identity provider.
  This may vary, depending on your SAML implementation, and on the configured SAML identity provider.
- Go to the platform sign in page, and verify that the page includes the option to sign in with SAML.
- Use the SAML option to sign in using valid SAML user credentials.
- You should be signed in with the user whose credentials you entered to authenticate, and you should be redirected to the platform dashboard.

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.

- Required credentials for installation:

```
db_user=test
db_pass=test
```

- Required variables for installation (for platform domain name use an ip or a domain name that resolves to your machine):

```
platform_domain_name=
db_name=platform
generate_self_signed_cert=true
```

Credentials and variables listed above need to be exported prior to continuing with the instructions: `export var_name=value` or they can be replaced with their values in any of the commands that contain them.

Load the Supervisor configuration

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

```
# Launch supervisord
systemctl start supervisord
```

- Check if the supervisord daemon is running:

```
# Check if supervisord is running
systemctl status supervisord
```

- Enable supervisord service:

```
# Enable supervisord service
systemctl enable supervisord
```

- Check the statuses of the tasks managed by Supervisor:

```
# Check tasks
supervisorctl status
```

Stop platform processes

- Platform processes need to be stopped before performing database and elasticsearch modifications:

```
# Stop platform processes
supervisorctl stop all
```

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. Create the database and load the default fixtures for the platform.

- Create the database schema:

```
# Create the database schema
EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/migrations/create-db.sh
```

- Generate the default platform fixtures:

```
# Generate the default platform fixtures
EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/migrations/load-fixtures.sh
```

Bootstrap Elasticsearch

To make sure you are applying the latest Elasticsearch schema, migrate Elasticsearch indices. The migration process is idempotent. It sets up and builds the required/specified indices with aliases and mappings, and it updates the index mapping templates, if necessary.

- Migrate Elasticsearch indices. The command runs in the background. In case of an SSH disconnection, the process should keep running normally. This upgrade action is idempotent. First, it tries to update the Elasticsearch index mappings in-place. If it is not possible, it proceeds to reindex the existing Elasticsearch indexes.

```
# Migrate Elasticsearch indices
yes | EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/eiq-platform search upgrade
```

Index migration log messages, if any, are printed to the terminal. If no messages are printed to the terminal, the process completed successfully.

After completing the index migration, you can validate Elasticsearch to verify that the indices are migrated successfully, and that Elasticsearch works normally.

- Validate the Elasticsearch index migration:

```
# Validate Elasticsearch index migration
EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/eiq-platform search validate
```

Bootstrap Neo4j

Create the graph schema

- Run the following command to create the graph schema and to apply the necessary graph database migrations to the Neo4j database:

```
# create the graph schema / apply graph database migrations
EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/eiq-platform graph upgrade
```

Restart Nginx

- Restart nginx:

```
# Restart nginx
systemctl restart nginx
```

- Check if nginx is running:

```
# Check if nginx is running
systemctl status nginx
```

Load the dashboard

To populate the platform dashboard, you need to fetch data from Elasticsearch and Kibana. elasticdump helps you do that.

- Load the platform dashboard:

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

Update hostname

- Use following commands to set hostname value to your platform domain name:

```
# Set default hostname
old_config=$(PGPASSWORD=$db_pass psql -h localhost -U $db_user -d $db_name -tWwc "select config
from configuration where name='server'")
new_config=$(echo $old_config | jq ".server_name = \"${platform_domain_name}\"")
PGPASSWORD=$db_pass psql -h localhost -U $db_user -d $db_name -tWwc "update configuration set
config='$new_config' where name='server'"
echo "You may need to set hostname manually. Please refer to the installation documentation."
```

- If you use different hostname to access the machine from outside, you will have to change it manually in system settings when you log in to platform.

Reload Supervisor configurations

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

```
# Restart supervisor
systemctl restart supervisord
```

- Check the statuses of the tasks managed by Supervisor. Re-run this command until you can confirm all services are running:

```
# Check supervisor tasks (re-run until you see that services are up)
supervisorctl status
```

Access the platform

Open firewall ports

- Open http/https ports on firewall. Here is how if you use `firewalld`:

```
# Open firewalld ports for http(s)
if [ "$(firewall-cmd --state)" == "running" ]; then firewall-cmd --permanent --zone=public --add-service=http --add-service=https; firewall-cmd --reload; fi
```

Reboot the machine

- Reboot the machine to make sure all services have picked up new settings and defaults:

```
# Reboot the machine
reboot &
```

Login to platform

- 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.

Install platform extensions

Install extensions to expand platform functionality, and to integrate it with external intel providers and data sources.

Danger:
Due to breaking changes, the following enrichers and incoming feeds are *not working* on EclecticIQ Platform release *1.14.4 (and earlier) until release 2.0.2 included*:

- Enrichers:
  - FireEye iSIGHT
  - Intel 471
  - Recorded Future
- Incoming feeds:
  - BFK API
  - Cisco Threat Grid Curated Feed
  - Cisco Threat Grid Samples API
  - Crowdstrike Falcon Intelligence Indicator Feed
  - Crowdstrike Falcon Intelligence Reports Feed
  - Crowdstrike Falcon Intelligence Threat Actor Feed
  - FireEye iSIGHT Intelligence Report API

To restore full functionality for these enrichers and incoming feeds, upgrade to EclecticIQ Platform release *2.1.0* or later.

EclecticIQ Platform ships with a set of built-in extensions, such as incoming feeds and enrichers, to expand platform functionality, and to enable interoperability with external systems, intel providers, and data sources.

You can download and install more extensions to add incoming and outgoing feed transport and content types, as well as enrichers. This modular approach enables you to implement as many, or as few, integrations as you need, based on your organization requirements and goals.

Currently, the following extensions are available for download and installation:

Extension	Download	Type
Anubis	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	incoming feed transport and content types
ArcSight	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	outgoing feed content type
BFK	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	incoming feed transport and content types
CAPEC	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	incoming feed content type

Extension	Download	Type
Censys	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	enricher
CIRCL	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	enricher
Core — INSTALL ME FIRST	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	incoming and outgoing feed transport and content types
Crowdstrike	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	incoming feed, enricher
CVE Search	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	incoming feed transport and content types, enricher
DomainTools	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	enricher
Farsight	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	enricher
FireEye	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	incoming feed transport type, enricher
Flashpoint	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	enricher
Fox-IT	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	enricher
Intel 471	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	incoming feed, enricher
OpenPhish	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	incoming feed transport and content types
OpenSource	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	-
PassiveTotal	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	enricher
PhishMe	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	incoming feed transport and content types
Recorded Future	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	enricher
Reports	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	outgoing feed content type
S3	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	incoming and outgoing feed transport type
Shodan	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	enricher

Extension	Download	Type
Sighting	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	incoming feed transport type
SpyCloud	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	incoming feed transport and content types, enricher
STIX TAXII	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	incoming and outgoing feed transport and content types
Threat Grid	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	ncoming feed transport and content types
Threat Recon	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	incoming feed content type
VirusTotal	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	enricher
Wapack Labs	**https://downloads.eclecticiq.com/Extensions/** (https://downloads.eclecticiq.com/extensions/)	incoming feed transport type

Download the extension

- Go to the EclecticIQ Platform Extensions repository at https://downloads.eclecticiq.com/Extensions/.
- Browse to the extension you want to install, and click its name.
- On the extension page, **General** tab, browse to the **Downloads** section, and then click the extension package name to download it.
  Extension package name format: *eclecticiq-extension-{extension_name}-{platform_release}.tar.gz*
  Example:
  - *eclecticiq-extension-anubis-1.0.tar.gz*: Anubis Cyberfeed incoming feed for EclecticIQ Platform release 1.14.x.
  - *eclecticiq-extension-anubis-2.0.tar.gz*: Anubis Cyberfeed incoming feed for EclecticIQ Platform release 2.0.x.
- Save the extension package to a local directory.

Switch to eclecticiq

- Switch to the `eclecticiq` user by running the following command(s):

```
$ su - eclecticiq
```

Activate venv

- Activate a Python virtual environment:

```
$ source /opt/eclecticiq/platform/api/bin/activate
```

- Export the platform settings configuration to create the necessary environment variables:

```
(api) $ export EIQ_PLATFORM_SETTINGS="/opt/eclecticiq/etc/eclecticiq/platform_settings.py"
```

Install the extension

- Install the downloaded extension with **pip install** (https://pip.pypa.io/en/stable/reference/pip_install/):

```
$ pip install eclecticiq-extension-${extension-name}-${platform-release}.tar.gz

# Example:
$ pip install eclecticiq-extension-anubis-2.1.0.tar.gz
```

Reload Supervisor configurations

Warning:
When you edit or update Supervisor configurations, run `systemctl restart supervisor` and `supervisorctl reload`, so that Supervisor can pick up and reload any updated configurations to the platform with the latest changes.

After editing Supervisor-managed configuration (*.ini*) files restart, and then reload Supervisor, so that it can pick up all changes and apply them:

```
$ systemctl restart supervisord

$ supervisorctl reload
```

This applies to all Supervisor-managed programs, applications, extensions, and services in the platform.

Load the fixtures

- As a final step, load the fixtures to complete the installation, and to bootstrap the extension:

```
(api) $ eiq-platform database load-fixtures
```

The extension is ready for use in the platform.

Remove an extension

You may need to remove an extension for several reasons: to upgrade it to a newer release, to reinstall it in order to address functionality issues, or because you do not need it any longer.

To remove an extension, run the standard `pip uninstall` command, and then specify only the name of the extension, without platform release number, and without extension archive file type.

```
$ pip uninstall eclecticiq-extension-${extension-name}

# Example: uninstall eclecticiq-extension-arcsight-2.0.tar.gz
$ pip uninstall eclecticiq-extension-arcsight
```

Extension naming	cheatsheet
Default naming format	`eclecticiq-extension-${extension-name}-${platform-release}.tar.gz`
Install extension	`eclecticiq-extension-${extension-name}-${platform-release}.tar.gz`
Uninstall extension	`eclecticiq-extension-${extension-name}`

Create custom extensions with the SDK

You can extend EclecticIQ Platform functionality — for example, by integrating the platform with an external system such as a SIEM, or by hooking it up to a custom intel provider service. EclecticIQ Platform SDK is the companion development kit to create integration extensions, such as feeds and enrichers.

The SDK provides a structured framework to build your extensions, to implement transport and content types for custom feeds, as well as to create new enrichers, based on your organization needs and requirements.

The SDK includes documentation and examples to help you along the way.
The documentation is included in the downloadable SDK archive file. To open it:

- Decompress the SDK archive to a local working directory.
- Browse to the `/docs` sub-directory.

- Open the `index.html` index file in a web browser to access the SDK documentation and the main navigation.

In the `/examples` sub-directory you can find reference examples of:

- A transformer, that is, a content type parser
- A provider, that is, a transport type handler
- An enricher.

> When you create a custom transformer to parse an input content type to JSON for ingestion into the platform, stress-test it to gauge its ingestion limit.
> For example, ingesting a package containing 10,000 entities may slow down the process to a crawl.
> Therefore, it is a good practice to feed transformers packages containing less than 10,000 entities.

API endpoints

EclecticIQ Platform includes a public API with a number of endpoints exposing services such as the authentication mechanism, as well as access to the platform assets and resources, such as entities, observables, enrichment tasks, and data sources.

The API accepts JSON requests and it returns JSON responses in UTF-8 encoding: `Content-Type:application/json; charset=utf-8`
The authentication mechanism is based on **JWT (JSON Web Tokens)** `(https://tools.ietf.org/html/rfc7519)`.

The public API methods and their arguments are documented, and you can familiarize yourself with the API calls and responses in a safe playground.
To access it, append `/swagger` to your platform instance base URL.
Example:

```
https://${my.platform.instance.org} + /swagger --> https://${my.platform.instance.org}/swagger
```

For reference, this overview sums up the public API endpoints:

```
{
  "data": {

    "/api/": [
      "GET",
      "HEAD",
      "OPTIONS"
    ],

    "/api/auth": [
      "OPTIONS",
      "POST"
    ],

    "/api/enrichers/": [
      "GET",
      "HEAD",
      "OPTIONS"
    ],

    "/api/enrichers/<int:id>": [
```

```
  "/api/enrichers/<int:id>": [
    "GET",
    "HEAD",
    "OPTIONS"
  ],

  "/api/enrichment-tasks/<uuid:id>": [
    "GET",
    "HEAD",
    "OPTIONS"
  ],

  "/api/entities/": [
    "GET",
    "HEAD",
    "OPTIONS"
  ],

  "/api/entities/<uuid:id>": [
    "GET",
    "HEAD",
    "OPTIONS"
  ],

  "/api/entities/<uuid:id>/enrich": [
    "OPTIONS",
    "POST"
  ],

  "/api/entities/<uuid:id>/enrichers": [
    "GET",
    "HEAD",
    "OPTIONS"
  ],

  "/api/observables/": [
    "GET",
    "HEAD",
    "OPTIONS",
    "POST"
  ],

  "/api/observables/<int:id>": [
    "GET",
    "HEAD",
    "OPTIONS",
    "PATCH"
  ],

  "/api/observables/<int:id>/enrich": [
    "OPTIONS",
    "POST"
  ],

  "/api/observables/<int:id>/enrichers": [
    "GET",
    "HEAD",
    "OPTIONS"
  ],

  "/api/sources/": [
    "GET",
    "HEAD",
    "OPTIONS"
  ],
```

```
    "/api/swagger.json": [
      "GET",
      "HEAD",
      "OPTIONS"
    ],

    "/api/users/self": [
      "GET",
      "HEAD",
      "OPTIONS"
    ]

  }
}
```

Before you upgrade

Before upgrading the platform to a newer release, it is a good idea to check the upgrade prerequisites and to back up platform data and configurations. This reference checklist provides guidelines to plan a smooth upgrade process.

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 successfully execute several commands in the command line or in the terminal, you may need root-level access rights.
To obtain admin rights, run the following command(s):

```
$ sudo su -
```

Alternatively:

- Grant admin rights to a specific user, who can then log in with their password to perform admin tasks:

```
$ su - ${username}
```

Or:

- Prefix `sudo` to the command you want to run:

```
$ sudo ${command}
```

The upgrade procedure consists of the following steps:

Before the upgrade

- Disable rules
- Exit the platform
- Back up your data
- Shut down the platform
- Check the prerequisites

During the upgrade

- Install the new release of the platform

After the upgrade

- Check the configuration
- Check third-party configurations
- Run a final check
- Reload Supervisor configurations

- Install extensions

Before the upgrade

Disable rules

Disable all platform rules: entity, observable, enrichment, and discovery rules.

You can disable rules in one of the following ways:

In the rule detail pane

- Click **Data configuration > Rules > Observable** , or **Data configuration > Rules > Entity** , or **Data configuration > Rules > Enrichment**, or **Data configuration > Rules > Discovery** to display the observable, entity, enrichment, or discovery rule overview.
- On the rule overview click anywhere on the row corresponding to the rule you want to disable.
- On the rule detail panel:
  - Click **Actions > Disable** to disable the rule.

    Alternatively:
  - On the **Details** tab click **Disable**.

A notification message is displayed to confirm the change.

In the rule overview

- Click **Data configuration > Rules > Observable** , or **Data configuration > Rules > Entity** , or **Data configuration > Rules > Enrichment**, or **Data configuration > Rules > Discovery** to display the observable, entity, enrichment, or discovery rule overview.
- On the rule overview click the ⋮ menu on the row corresponding to the rule you want to disable.
- From the drop-down menu select **Disable** to disable the rule.

A notification message is displayed to confirm the change.

Bulk disable

- Click **Data configuration > Rules > Observable** , or **Data configuration > Rules > Entity** , or **Data configuration > Rules > Enrichment**, or **Data configuration > Rules > Discovery** to display the observable, entity, enrichment, or discovery rule overview.
- On the top-left corner click the quick filter icon ( ) to display the available rule quick filters.
- Click **Show**, select **Enabled**, and then click **OK** to display only enabled rules.
- To select all the rules on the view click the checkbox on the top-left corner of the table.
- To disable all the selected rules in bulk, on the quick filter horizontal bar click **Actions > Disable**.

A notification message is displayed to confirm the change.

Exit the platform

Sign out of the platform:

- Click the active user profile image on the left-hand navigation sidebar, bottom-left 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 graciously shut down EclecticIQ Platform, stop all platform-related services and processes.

- A normal/standard platform shutdown does not involve any specific procedure or step sequence.
- If you shut down the platform before performing a platform upgrade , follow the steps described under **Shutdown before a platform upgrade**.
  In this case, it is important that you stop platform components, services, and processes gradually to avoid hanging queues, tasks or PIDs.

Normal shutdown

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

```
$ supervisorctl stop all
```

```
$ systemctl stop postgresql-10 redis neo4j kibana logstash elasticsearch postfix
```

Shutdown before a platform upgrade

If you are shutting down the platform before performing an *upgrade* or a *database backup*, stop platform components in the order described below to make sure no Celery tasks are left over in the queue, and no read/write activity is in progress on Redis.
This prevents hanging tasks in the queue from interfering with the upgrade or backup procedures.

- Stop *platform-api*:

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

- Stop the Celery beat:

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

- Check Celery queues; they should be emtpy:

```
# Launch redis-cli
$ redis-cli

$ > llen enrichers

$ > llen integrations

$ > llen priority_enrichers

$ > llen priority_providers

$ > llen priority_utilities

$ > llen providers

$ > llen reindexing

$ > llen utilities
```

- To delete a non-empty Celery queue, run the following command(s):

```
# Launch redis-cli
$ redis-cli

# Delete the entity ingestion queue
$ > del "queue:ingestion:inbound"

# Delete the graph ingestion queue
$ > del "queue:graph:inbound"

# Delete the search indexing queue
$ > del "queue:search:inbound"
```

- Stop the remaining Celery workers:

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

- Stop Supervisor-managed workers:
  - *intel-ingestion*
  - *search-ingestion*
  - *graph-ingestion*
  - *opentaxii*
  - *neo4j-batching*

```
$ supervisorctl stop all
```

Check that there are no leftover PID files:

- First, make sure that no platform-related PID is running:

```
$ ps aux | grep beat
```

- If any platform-related PIDs are running, terminate them with the `kill` command.
- Manually remove any leftover PID files with the `rm` command. Usually, PID files are stored in */var/run*.

Check the prerequisites

> When upgrading dependencies and third-party components, refer to their official documentation for detailed instructions on installation and upgrade procedures, and look up their official release notes for any product changes that may impact your environment.

Before upgrading to a new platform release, **check prerequisites and dependencies**, and verify that all required third-party components are already installed on the target system, and their versions match the recommended version information provided.

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 you upgraded from.

Core platform configuration files	
platform_settings.py	Contains core platform settings like security key value, authentication bearer token expiration time, URLs ponting to external components Celery-managed tasks, and LDAP configuration.
opentaxii.yml	Contains **OpenTAXII** `(https://opentaxii.readthedocs.io/)` configuration parameters like URL and port for the service, as well as the designated inbound queue and message broker to use.

Verify that the platform configuration files reflect the new, upgraded environment.

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 the configurations of third-party components and dependencies.

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/).

After the upgrade

About proxy settings

If the platform is configured to be behind a proxy, make sure the platform proxy configuration allows bypassing local hosts *localhost* and *127.0.0.1*.

If you cannot access the platform proxy settings, and if terminal commands acting on platform resources fail to execute correctly, prepend `NO_PROXY=127.0.0.1` to the commands to bypass the proxy server on the fly.

This behavior may affect the following commands:

```
# Start Python shell
$ /opt/eclecticiq/platform/api/bin/eiq-platform shell

# Request current db version
$ /opt/eclecticiq/platform/api/bin/eiq-platform database current-version

# Migrate Elasticsearch indices
$ /opt/eclecticiq/platform/api/bin/eiq-platform search upgrade

# Validate Elasticsearch index migration
$ /opt/eclecticiq/platform/api/bin/eiq-platform search validate

# Reindex Neo4j graph db
$ /opt/eclecticiq/platform/api/bin/eiq-platform graph reindex

# Migrate Neo4j graph db
$ /opt/eclecticiq/platform/api/bin/eiq-platform graph upgrade

# Load fixtures
$ /opt/eclecticiq/platform/api/bin/eiq-platform database load-fixtures

# Registered and enabled extension state validation
$ /opt/eclecticiq/platform/api/bin/eiq-platform extensions validate
```

Example:

```
$ NO_PROXY=127.0.0.1 /opt/eclecticiq/platform/api/bin/eiq-platform graph upgrade
```

About Elasticsearch indexes

If you need to prioritize migrating Elasticsearch indexes, process at least the following ones:

- *stix*: indexes entities
- *extracts*: indexes observables

Elasticsearch index name	Description
audit	Records audit trail events related to entities, datasets, enrichers, incoming and outgoing feeds, rules and tasks.
documents	Records log information related to ingestion, tasks, and task scheduling.
*draft-entities-**	Indexes draft entity data, that is, entities that are currently saved as drafts, and that have not yet been published to the platform. These entities are not searchable in the platform.
*extracts-**	Indexes all observable data.
*logstash-**	Indexes log aggregation and logging information such as host, http request types, http response status codes, platform component, and path to the log where log entires are saved to.
*statsd-**	Collects metrics about received packets and detected invalid or not well-formed lines in the ingested packets.
*stix**	Indexes published entity data, that is, entities that are published to the platform. These entities are searchable in the platform.

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
- To check if a core service is enabled to start at system bootup run the following command(s):

```
systemctl is-enabled ${service_name}
```

- To check if a core service is running run the following command(s):

```
systemctl status ${service_name}
```

- To start a core service run the following command(s):

```
systemctl start ${service_name}
```

Check core processes and services

Nginx

- Verify that Nginx is up and running by checking the web server status:

```
$ systemctl status nginx
```

Supervisor

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

```
systemctl status supervisord
```

PostgreSQL

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

```
systemctl status postgresql-10
```

or:

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

Check search indexing and graph

Elasticsearch

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

```
systemctl status elasticsearch
```

Neo4j

To check if Neo4j is running, run the following command(s):

```
$ systemctl status neo4j
```

Check search indexing and graph 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
# HTTP port: 7474; HTTPS port: 7473
$ curl localhost:7474
```

Re-enable and run the rules

Before starting Supervisor-managed ingestion processes, enable again the rules you previously disabled.
Run the re-enabled rules after completing the data migration, so that they can filter out any observables marked to be ignored.

Enable platform rules: entity, observable, enrichment, and discovery rules.

You can enable rules in one of the following ways:

In the rule detail pane

- Click **Data configuration > Rules > Observable** , or **Data configuration > Rules > Entity** , or **Data configuration > Rules > Enrichment**, or **Data configuration > Rules > Discovery** to display the observable, entity, enrichment, or discovery rule overview.
- On the rule overview click anywhere on the row corresponding to the rule you want to enable.
- On the rule detail panel:
  - Click **Actions > Enable** to enable the rule.

    Alternatively:
  - On the **Details** tab click **Enable**.

A notification message is displayed to confirm the change.

In the rule overview

- Click **Data configuration > Rules > Observable** , or **Data configuration > Rules > Entity** , or **Data configuration > Rules > Enrichment**, or **Data configuration > Rules > Discovery** to display the observable, entity, enrichment, or discovery rule overview.
- On the rule overview click the ⋮ menu on the row corresponding to the rule you want to enable.
- From the drop-down menu select **Enable** to enable the rule.

A notification message is displayed to confirm the change.

Bulk enable

- Click **Data configuration > Rules > Observable** , or **Data configuration > Rules > Entity** , or **Data configuration > Rules > Enrichment**, or **Data configuration > Rules > Discovery** to display the observable, entity, enrichment, or discovery rule overview.
- On the top-left corner click the quick filter icon ( ) to display the available rule quick filters.

- Click **Show**, select **Disabled**, and then click **OK** to display only disabled rules.
- To select all the rules on the view click the checkbox on the top-left corner of the table.
- To enable all the selected rules in bulk, on the quick filter horizontal bar click **Actions > Enable**.

A notification message is displayed to confirm the change.

Reload Supervisor configurations

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 19527, uptime 0:00:03
intel-ingestion:0                 RUNNING   pid 19071, uptime 0:00:51
intel-ingestion:1                 RUNNING   pid 19070, uptime 0:00:51
intel-ingestion:2                 RUNNING   pid 19073, uptime 0:00:51
intel-ingestion:3                 RUNNING   pid 19072, uptime 0:00:51
neo4j-batching                    RUNNING   pid 19268, uptime 0:00:43
opentaxii                         RUNNING   pid 19330, uptime 0:00:36
platform-api                      RUNNING   pid 19077, uptime 0:00:51
search-ingestion                  RUNNING   pid 19075, uptime 0:00:51
task:beat                         RUNNING   pid 19061, uptime 0:00:51
task:discovery                    RUNNING   pid 19068, uptime 0:00:51
task:discovery-priority           RUNNING   pid 19065, uptime 0:00:51
task:enrichers                    RUNNING   pid 19056, uptime 0:00:51
task:enrichers-priority           RUNNING   pid 19062, uptime 0:00:51
task:entity-rules-priority        RUNNING   pid 19063, uptime 0:00:51
task:extract-rules-priority       RUNNING   pid 19055, uptime 0:00:51
task:incoming-transports          RUNNING   pid 19053, uptime 0:00:51
task:incoming-transports-priority RUNNING   pid 19054, uptime 0:00:51
task:outgoing-feeds               RUNNING   pid 19066, uptime 0:00:51
task:outgoing-feeds-priority      RUNNING   pid 19057, uptime 0:00:51
task:outgoing-transports          RUNNING   pid 19060, uptime 0:00:51
task:outgoing-transports-priority RUNNING   pid 19058, uptime 0:00:51
task:reindexing                   RUNNING   pid 19064, uptime 0:00:51
task:utilities                    RUNNING   pid 19059, uptime 0:00:51
task:utilities-priority           RUNNING   pid 19067, uptime 0:00:51
```

Install extensions

After successfully completing the platform upgrade, you can proceed to install extensions as necessary to expand platform functionality, and to add support for a broad range of transport types and content types for incoming and outgoing feeds, as well as many enrichers.

Warning:
To avoid compatibility issues while doing a major library upgrade or change, we recommend rebuilding virtualenv. We also recommend relying on automatic OS upgrades instead of rebuilding the packages manually.

Platform maintenance upgrade

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

This maintenance upgrade guide is intended to be used for upgrades from 2.1.x versions. If you have platform version prior to 2.1.0, please use Centos upgrade guide.

- If you are using a fresh VM, refer to **Create the YUM repository configuration files** section in Centos upgrade guide first. Virtual machines are not distributed with YUM repositories by default.
- Stop all platform processes before continuing:

```
# Stop supervisor
systemctl stop supervisord
```

Install the upgraded version of EclecticIQ Platform

- Back up the existing platform settings:

```
# Back up platform settings
\cp /opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/etc/eclecticiq/platform_settings.py.backup
```

- Rename dependency repo name from `platform-centos-deps-2.1` to `platform-centos-deps` if you haven't already done so:

```
# Rename dependency repo name
sed -ri "s/platform-centos-deps-2.1/platform-centos-deps/g" /etc/yum.repos.d/platform-centos-
deps.repo
```

- Install the upgraded version of the platform:

```
# Install new version of platform
yes | yum -y --disablerepo=* --enablerepo="platform-centos-deps,platform-centos" upgrade
eclecticiq-platform
```

- If your package manager is configured not to overwrite existing configuration files, you may need to replace the previous platform configuration file with the new one provided with the current platform release:

```
# Use new platform settings
\mv /opt/eclecticiq/etc/eclecticiq/platform_settings.py.rpmnew
/opt/eclecticiq/etc/eclecticiq/platform_settings.py || true
```

- Install the upgraded version of the platform documentation:

```
# Install new version of docs
yes | yum -y --disablerepo=* --enablerepo="platform-centos-deps,platform-centos" upgrade
eclecticiq-platform-docs
```

- Copy your existing PostgreSQL settings to the new platform configuration file:

```
# Copy SQLALCHEMY_DATABASE_URI value
sed -i '/^\s*SQLALCHEMY_DATABASE_URI\s*=.*/d' /opt/eclecticiq/etc/eclecticiq/platform_settings.py
cat /opt/eclecticiq/etc/eclecticiq/platform_settings.py.backup | grep
'^\s*SQLALCHEMY_DATABASE_URI\s*=' >> /opt/eclecticiq/etc/eclecticiq/platform_settings.py
```

- Copy your existing secret key:

```
# Copy SECRET_KEY value
sed -i '/^\s*SECRET_KEY\s*=.*/d' /opt/eclecticiq/etc/eclecticiq/platform_settings.py
cat /opt/eclecticiq/etc/eclecticiq/platform_settings.py.backup | grep '^\s*SECRET_KEY\s*=' >>
/opt/eclecticiq/etc/eclecticiq/platform_settings.py
```

- Compare the new and the previous versions of the configuration files:

```
# Diff config files
diff /opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/etc/eclecticiq/platform_settings.py.backup
```

Review, identify, and update any custom variables that you have previously set manually in the platform configuration file.

Upgrade third-party dependencies

- Run a YUM upgrade:

```
# Upgrade all third-party dependencies
yum -y --disablerepo=* --enablerepo="platform-centos-deps" upgrade
```

Migrate PostgreSQL

```
# Migrate PostgreSQL
EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/migrations/db-migration.sh
```

Migrate the Elasticsearch indices

```
# Migrate Elasticsearch indexes
yes | /opt/eclecticiq/platform/api/bin/eiq-platform search upgrade
```

Migrate the graph database

```
# Migrate the graph database
EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/eiq-platform graph upgrade
```

Run the fixtures

```
# Generate default fixtures
EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/migrations/load-fixtures.sh
```

Reboot the system

```
# Reboot the system to make all configuration changes effective
reboot &
```

Upgrade the platform

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

- If you are using a fresh VM, refer to **Create the YUM repository configuration files** section in Centos upgrade guide first. Virtual machines are not distributed with YUM repositories by default.
- Required credentials for the platform installation:

```
eiq_user=
eiq_pass=
```

- Required variables for the platform installation:

```
data_dir=/media
platform_repo_path=downloads.eclecticiq.com/platform-centos/2.1
platform_deps_path=downloads.eclecticiq.com/platform-centos-deps/2.1
binary_repo_path=downloads.eclecticiq.com/platform-binary-deps
```

Credentials and variables listed above need to be exported prior to continuing with the instructions: `export var_name=value` or they can be replaced with their values in any of the commands that contain them.

- Stop all platform processes before continuing:

```
# Stop supervisor
systemctl stop supervisord
```

Create the YUM repository configuration files

- Create a file with the repository information for the dependencies:

```
# Create and populate yum repo file for platform dependencies
echo "
[platform-centos-deps]
name=platform-centos-deps
baseurl=https://${platform_deps_path}
gpgcheck=0
repo_gpgcheck=0
enabled=1
username=${eiq_user}
password=${eiq_pass}
gpgkey=https://downloads.eclecticiq.com/public/GPG-KEY-eclecticiq
       https://downloads.eclecticiq.com/public/GPG-KEY-elasticsearch
       https://downloads.eclecticiq.com/public/neo4j.gpg.key
       https://downloads.eclecticiq.com/public/nginx_signing.key
       https://downloads.eclecticiq.com/public/IUS-COMMUNITY-GPG-KEY
       https://downloads.eclecticiq.com/public/RPM-GPG-KEY-EPEL-7
       https://downloads.eclecticiq.com/public/NODESOURCE-GPG-SIGNING-KEY-EL.key
priority=1
" > /etc/yum.repos.d/platform-centos-deps.repo
```

Update and set user name and password as needed, based on your system environment.

- Create a file with the repository information for the platform:

```
# Create and populate yum repo file for EclecticIQ Platform
echo "
[platform-centos]
name=platform-centos
baseurl=https://${platform_repo_path}
gpgcheck=1
repo_gpgcheck=0
enabled=1
username=${eiq_user}
password=${eiq_pass}
gpgkey=https://downloads.eclecticiq.com/public/GPG-KEY-eclecticiq
" > /etc/yum.repos.d/platform-centos.repo
```

Update and set user name and password as needed, based on your system environment.

- Clear the YUM version lock:

```
# Clear version lock
yum versionlock clear
```

Install the upgraded version of EclecticIQ Platform

- Back up the existing platform settings:

```
# Back up platform settings
\cp /opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/etc/eclecticiq/platform_settings.py.backup
```

- Install the upgraded version of the platform:

```
# Install new version of platform
yes | yum -y --disablerepo=* --enablerepo="platform-centos-deps,platform-centos" upgrade
eclecticiq-platform
```

- If your package manager is configured not to overwrite existing configuration files, you may need to replace the previous platform configuration file with the new one provided with the current platform release:

```
# Use new platform settings
\mv /opt/eclecticiq/etc/eclecticiq/platform_settings.py.rpmnew
/opt/eclecticiq/etc/eclecticiq/platform_settings.py || true
```

- Install the upgraded version of the platform documentation:

```
# Install new version of docs
yes | yum -y --disablerepo=* --enablerepo="platform-centos-deps,platform-centos" upgrade
eclecticiq-platform-docs
```

- Copy your existing PostgreSQL settings to the new platform configuration file:

```
# Copy SQLALCHEMY_DATABASE_URI value
sed -i '/^\s*SQLALCHEMY_DATABASE_URI\s*=.*/d' /opt/eclecticiq/etc/eclecticiq/platform_settings.py
cat /opt/eclecticiq/etc/eclecticiq/platform_settings.py.backup | grep
'^\s*SQLALCHEMY_DATABASE_URI\s*=' >> /opt/eclecticiq/etc/eclecticiq/platform_settings.py
```

- Copy your existing secret key:

```
# Copy SECRET_KEY value
sed -i '/^\s*SECRET_KEY\s*=.*/d' /opt/eclecticiq/etc/eclecticiq/platform_settings.py
cat /opt/eclecticiq/etc/eclecticiq/platform_settings.py.backup | grep '^\s*SECRET_KEY\s*=' >>
/opt/eclecticiq/etc/eclecticiq/platform_settings.py
```

- Compare the new and the previous versions of the configuration files:

```
# Diff config files
diff /opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/etc/eclecticiq/platform_settings.py.backup
```

Review, identify, and update any custom variables that you have previously set manually in the platform configuration file.

Upgrade PostgreSQL

- Dump the PostgreSQL 9.x database. The resulting dump file may be very large (several GBs); therefore, make sure you have plenty of free disk space to save it. The code example provided here dumps the database to the current directory; change it to a different one, if needed:

```
# Dump database
sudo -u postgres -i pg_dumpall > dump.sql
```

- Install PostgreSQL 10 client and the repo keys:

```
# Install PostgreSQL client and keys
yum -y --disablerepo=* --enablerepo="platform-centos-deps,epel,base" install postgresql10-10.1
```

- Install PostgreSQL server and the additional components:

```
# Install PostgreSQL server
yum -y --disablerepo=* --enablerepo="platform-centos-deps,epel,base" install postgresql10-devel
postgresql10-contrib postgresql10-server postgresql10-libs
```

- Pin the following packages, so that they stay on the corresponding current versions, and they are not automatically updated:

```
# Pin package versions
yum versionlock postgresql10-server postgresql10-contrib postgresql10-devel
```

- Stop PostgreSQL 9.5:

```
# Stop postgresql-9.5
systemctl stop postgresql-9.5
```

- Disable PostgreSQL 9.5:

```
# Disable postgresql-9.5
systemctl disable postgresql-9.5
```

- Initialize the PostgreSQL 10 database:

```
# Initialize the database
/usr/pgsql-10/bin/postgresql-10-setup initdb
```

- If you moved the `9.5` directory from `/var/lib/pgsql` to your data dir, and if you created a symlink to it, repeat the same step with the `10` directory. We recommend moving the whole `pgsql` dir, so not just the `9.5` subdir, and symlinking it, so that data for any new PostgreSQL installs is automatically stored to your data dir. If you have already carried out this task, ignore this point and continue with the following steps.
- Start PostgreSQL 10:

```
# Start postgresql-10
systemctl start postgresql-10
```

- Enable PostgreSQL 10:

```
# Enable postgresql-10
systemctl enable postgresql-10
```

- Import the data:

```
# Import database data
sudo -u postgres -i psql < dump.sql
```

- Configure the authentication methods:

```
# Configure postgres auth
db_user=$(cat /opt/eclecticiq/etc/eclecticiq/platform_settings.py.backup | grep
SQLALCHEMY_DATABASE_URI | awk -F":" '{print $2}' | cut -c 3-)
sed -ri 's/^((host|local)\s.*)/#\1/g' /var/lib/pgsql/10/data/pg_hba.conf
echo "local all all peer" >> /var/lib/pgsql/10/data/pg_hba.conf
echo "local all ${db_user} trust" >> /var/lib/pgsql/10/data/pg_hba.conf
echo "host all all samenet md5" >> /var/lib/pgsql/10/data/pg_hba.conf
```

- Restart PostgreSQL 10:

```
# Restart postgresql-10
systemctl restart postgresql-10
```

Upgrade Elasticsearch

- Stop Elasticsearch:

```
# Stop Elasticsearch
systemctl stop elasticsearch
```

- Back up the Elasticsearch configuration:

```
# Back up Elasticsearch configuration:
\cp /etc/elasticsearch/elasticsearch.yml /etc/elasticsearch/elasticsearch.yml.$(date +%s).backup
```

- Uninstall *elasticdump*:

```
# Uninstall elasticdump
npm uninstall -g elasticdump
```

- Remove the *delete-by-query* plugin:

```
# Remove delete-by-query plugin
/usr/share/elasticsearch/bin/plugin remove delete-by-query
```

- Remove Elasticsearch:

```
# Remove elasticsearch
yum -y remove elasticsearch
```

- Install Elasticsearch version 5.6:

```
# Install new version of Elasticsearch
yes '' | yum -y --disablerepo=* --enablerepo="platform-centos-deps,epel,base" install
elasticsearch
```

- Pin the new version of Elasticsearch:

```
# Version-lock Elasticsearch
yum versionlock elasticsearch
```

- Copy the configuration files provided with the platform:

```
# Copy Elasticsearch configuration
\cp -R /opt/eclecticiq/etc-extras/elasticsearch/* /etc/elasticsearch/
```

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

```
# Set Elasticsearch data dir
sed -i 's|^\s*path\.data:.*|path.data: '${data_dir}'/elasticsearch|g'
/etc/elasticsearch/elasticsearch.yml
```

- Review and make additional changes to the Elasticsearch configuration as needed, based on your system environment.
- Reload the Elasticsearch *systemd* configuration:

```
# Reload systemd daemon
systemctl daemon-reload
```

- Modify the directory permissions for Elasticsearch:

```
# Modify directory permissions for Elasticsearch
chown -R elasticsearch:elasticsearch ${data_dir}/elasticsearch/
```

- Restart Elasticsearch:

```
# Restart Elasticsearch
systemctl restart elasticsearch
```

- Enable the Elasticsearch service to automatically start at system boot:

```
# Enable elasticsearch service
systemctl enable elasticsearch
```

- Install *elasticdump 3.0.0*:

```
# Install elasticdump globally
escaped_username=$(echo $eiq_user | sed 's|\@|%40|g')
npm install -g https://${escaped_username}:${eiq_pass}@${binary_repo_path}/elasticdump-3.0.0.tgz
```

Upgrade Neo4j

- Stop the current Neo4j instance:

```
# Stop Neo4j
systemctl stop neo4j
```

- Uninstall the current version of Neo4j:

```
# Remove Neo4j
yum -y remove neo4j
```

- Remove any custom Neo4j systemd files:

```
# Remove neo4j systemd file
rm -rf /etc/systemd/system/neo4j.service || true
```

- Create a database subdirectory in the Neo4j data dir:

```
# Neo4j directory structure
mkdir -p ${data_dir}/neo4j/data/databases
```

- Move `graph.db` to the newly created subdirectory:

```
# Move graph.db
\mv ${data_dir}/neo4j/graph.db ${data_dir}/neo4j/data/databases/
```

- Install the new Neo4j package:

```
# Install Neo4j
yes | yum -y --disablerepo=* --enablerepo="platform-centos-deps,epel,base" install neo4j
```

- Modify the directory permissions for Neo4j:

```
# Modify directory permissions for Neo4j
chown -R neo4j:neo4j ${data_dir}/neo4j
```

- Pin the new version of Neo4j:

```
# Pin Neo4j package
yum versionlock neo4j
```

- Copy the new Neo4j configuration file provided with the current platform release:

```
# Copy Neo4j config
\cp /opt/eclecticiq/etc-extras/neo4j/neo4j.conf /etc/neo4j/
```

- Set the Neo4j database location to your data directory:

```
# Set Neo4j database location
sed -ri 's|^(\s*dbms.directories.data\s*=).*|\1'${data_dir}/neo4j'|g' /etc/neo4j/neo4j.conf
```

- Enable the Neo4j service to automatically start at system boot:

```
# Enable Neo4j service
systemctl enable neo4j
```

- Restart Neo4j:

```
# Restart Neo4j
systemctl restart neo4j
```

- Set the Neo4j username and password in platform settings by editing `/opt/eclecticiq/etc/eclecticiq/platform_settings.py`, and by assigning appropriate values to `NEO4J_USER` and `NEO4J_PASSWORD`. If you have already set these values, copy them from the old platform settings file to the new one:

```
# Copy Neo4j password
sed -i '/^\s*NEO4J_PASSWORD\s*=.*/d' /opt/eclecticiq/etc/eclecticiq/platform_settings.py
cat /opt/eclecticiq/etc/eclecticiq/platform_settings.py.backup | grep '^\s*NEO4J_PASSWORD\s*=' >>
/opt/eclecticiq/etc/eclecticiq/platform_settings.py
```

- Repeat the same step with `NEO4J_USER`. By default, we set the default `neo4j` user; you can change it to a different user name, if needed:

```
# Set 'neo4j' user name
sed -i '/^\s*NEO4J_USER\s*=.*/d' /opt/eclecticiq/etc/eclecticiq/platform_settings.py
echo "NEO4J_USER = 'neo4j'" >> /opt/eclecticiq/etc/eclecticiq/platform_settings.py
```

- Set the initial password for the `neo4j` user. If `neo4j` is the designated user that should access the graph database, assign it the password you set in the platform settings file as the intial password:

```
# Set Neo4j password
NEO4J_PASSWORD=$(cat /opt/eclecticiq/etc/eclecticiq/platform_settings.py  | grep NEO4J_PASSWORD |
awk -F"'" '{print $2}')
neo4j-admin set-initial-password $NEO4J_PASSWORD
```

- If you are using a different user than `neo4j` to access the graph database, configure this user through the Neo4J web client. For further dteasils, see the Neo4J operations manual.

Upgrade Java JDK

- Download and unpack the new Java JDK version:

```
# Upgrade Oracle Java JDK 8
wget --user "${eiq_user}" --password "${eiq_pass}" -qO- https://${binary_repo_path}/jdk-8u151-
linux-x64.tar.gz | tar -xzf - -C /opt/
```

- Set Oracle Java JDK as the default Java:

```
# Set Oracle Java JDK as default
update-alternatives --install /usr/bin/java java /opt/jdk1.8.0_151/bin/java 100
update-alternatives --install /usr/bin/javac javac /opt/jdk1.8.0_151/bin/javac 100
update-alternatives --set java /opt/jdk1.8.0_151/bin/java
update-alternatives --set javac /opt/jdk1.8.0_151/bin/javac
```

- Verify the Java JDK version installed on the target system; run `java -version`, which should return this expected output:

```
java version "1.8.0_151"
Java(TM) SE Runtime Environment (build 1.8.0_151-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)
```

- Set up a profile file for Java JDK, where you define the `JAVA_HOME` environment variable:

```
# Export Java home
echo 'export JAVA_HOME=/opt/jdk1.8.0_151' > /etc/profile.d/jdk.sh
```

Upgrade third-party dependencies

- Run a YUM upgrade:

```
# Upgrade all third-party dependencies
yum -y --disablerepo=* --enablerepo="platform-centos-deps" upgrade
```

- Enable the Logstash service, as it may be disabled after the upgrade:

```
# Enable Logstash
systemctl enable logstash
```

Update the Nginx settings

- Back up the existing Nginx settings:

```
# backup nginx settings
\cp /etc/nginx/conf.d/platform.conf /etc/nginx/conf.d/platform.conf.backup
```

- Copy the new Nginx settings file provided with the current platform release:

```
# Copy Nginx settings
\cp /opt/eclecticiq/etc/nginx/conf.d/platform.conf /etc/nginx/conf.d/platform.conf
```

- Make sure that the SSL certificate paths point to the correct locations where your certificates are stored. You may copy this path from your old settings:

```
# Set Nginx certificate paths
sed -ri '/^\s*ssl_certificate(\s|_key\s).*/d' /etc/nginx/conf.d/platform.conf
cat /etc/nginx/conf.d/platform.conf.backup | grep -E '^\s*ssl_certificate(\s|_key\s)' >>
/etc/nginx/conf.d/platform.conf
```

- Restart Nginx:

```
# Restart Nginx
systemctl restart nginx
```

Migrate PostgreSQL

```
# Migrate PostgreSQL
EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/migrations/db-migration.sh
```

Migrate the Elasticsearch indices

```
# Migrate Elasticsearch indexes
yes | /opt/eclecticiq/platform/api/bin/eiq-platform search upgrade
```

Migrate the graph database

```
# Migrate the graph database
EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/eiq-platform graph upgrade
```

Run the fixtures

```
# Generate default fixtures
EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/migrations/load-fixtures.sh
```

Reboot the system

```
# Reboot the system to make all configuration changes effective
reboot &
```

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 on the PostgreSQL, Elasticsearch or Neo4j databases;
- No pending queues: this means that no read/write activity is in progress on the Redis database;
- No running Celery tasks;
- The platform is not running.

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

- Configuration files
- Databases
- Any SSL keys associated with the platform and its dependencies.

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

To successfully execute several commands in the command line or in the terminal, you may need root-level access rights.
To obtain admin rights, run the following command(s):

```
$ sudo su -
```

Alternatively:

- Grant admin rights to a specific user, who can then log in with their password to perform admin tasks:

```
$ su - ${username}
```

Or:

- Prefix `sudo` to the command you want to run:

```
$ sudo ${command}
```

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 template config files to use as boilerplates or scaffolding
$ 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

# Supervisor config file
/etc/supervisord.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

# Nginx web server config file
/etc/nginx/nginx.conf
/opt/eclecticiq/etc/nginx/conf.d/platform.conf

# Postfix email server config file
/etc/postfix/main.cf

# StasD config file
/etc/statsd/config.js # CentOS, RHEL
/opt/statsd/config.js # Ubuntu

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

# Elasticsearch config files
/etc/elasticsearch/elasticsearch.yml
/etc/elasticsearch/logging.yml

# 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 # CentOS, RHEL
/opt/eclecticiq/etc/default/logstash # Ubuntu

# Kibana config file
/etc/kibana/kibana.yml

# Redis message broker config file
/etc/redis.conf # CentOS, RHEL
/etc/redis/redis.conf # Ubuntu

# Neo4j config file
/etc/neo4j/neo4j.conf
```

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

Shut down the platform

If you are shutting down the platform before performing an *upgrade* or a *database backup*, stop platform components in the order described below to make sure no Celery tasks are left over in the queue, and no read/write activity is in progress on Redis.
This prevents hanging tasks in the queue from interfering with the upgrade or backup procedures.

- Stop *platform-api*:

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

- Stop the Celery beat:

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

- Check Celery queues; they should be emtpy:

```
# Launch redis-cli
$ redis-cli

$ > llen enrichers

$ > llen integrations

$ > llen priority_enrichers

$ > llen priority_providers

$ > llen priority_utilities

$ > llen providers

$ > llen reindexing

$ > llen utilities
```

- To delete a non-empty Celery queue, run the following command(s):

```
# Launch redis-cli
$ redis-cli

# Delete the entity ingestion queue
$ > del "queue:ingestion:inbound"

# Delete the graph ingestion queue
$ > del "queue:graph:inbound"

# Delete the search indexing queue
$ > del "queue:search:inbound"
```

- Stop the remaining Celery workers:

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

- Stop Supervisor-managed workers:
  - *intel-ingestion*
  - *search-ingestion*
  - *graph-ingestion*
  - *opentaxii*
  - *neo4j-batching*

```
$ supervisorctl stop all
```

Check that there are no leftover PID files:

- First, make sure that no platform-related PID is running:

```
$ ps aux | grep beat
```

- If any platform-related PIDs are running, terminate them with the `kill` command.
- Manually remove any leftover PID files with the `rm` command. Usually, PID files are stored in */var/run*.

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-10/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
```

When restoring a backup copy to an empty cluster, set `postgres` as a user.
Make sure the selected user for the restore operation has superuser access rights.

An easy way to restore a database dump created with `pg_dumpall` is to load it to an empty cluster.
If you try to load the backup copy to an existing copy of the same database, the process may return error messages because it tries to create relations that already exist in the target database.

File system level backup

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

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

- Go to the PostgreSQL data directory, typically *../pgsql/${version}/data/* (CentOS and RHEL), or *../postgresql/${version}/main/* (Ubuntu):

```
$ cd /var/lib/pgsql/10/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/10/data/*
```

- Go to the PostgreSQL data directory where you want to restore the database data, typically *../pgsql/${version}/data/* (CentOS and RHEL), or *../postgresql/${version}/main/* (Ubuntu):

```
$ cd /var/lib/pgsql/10/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 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.

Elasticsearch database backup example

> ✔ Key and parameter names, as well as values in the code examples are dummy.
> Replace them with appropriate names and values, depending on your environment and system configuration.

Example of an Elasticsearch database backup procedure:

- Create a directory to save the Elasticsearch backup/snapshot to.
- Make sure the `elasticsearch` has read and write access to it.

```
$ cd /db-backup

# Create Elasticsearch backup dir
$ mkdir elasticsearch

# Make sure the 'elasticsearch' user can access it
$ chown elasticsearch:elasticsearch elasticsearch/
```

- Add the database backup path to the */etc/elasticsearch/elasticsearch.yml* Elasticsearch configuration file:

```
path.repo: ["/db-backup/elasticsearch"]
```

- Restart Elasticsearch:

```
$ systemctl restart elasticsearch
```

- Define a repository for the backup:

```
$ curl -X PUT "http://localhost:9200/_snapshot/backup"
       -H "Content-Type: application/json"
       -d '{"type": "fs","settings":{"location":"/db-backup/elasticsearch"}}'

# copy-paste version:
$ curl -X PUT "http://localhost:9200/_snapshot/backup" -H "Content-Type: application/json" -d
'{"type": "fs","settings":{"location":"/db-backup/elasticsearch"}}'

# Example of a successful response
{"acknowledged": true}
```

- Verify that the newly created repository is correctly defined:

```
$ curl -X GET "localhost:9200/_snapshot/_all?pretty=true"
{
  "backup" : {
    "type" : "fs",
    "settings" : {
      "location" : "/db-backup/elasticsearch"
    }
  }
}
```

- Make a snapshot of the Elasticsearch database:

```
$ curl -X PUT "http://localhost:9200/_snapshot/backup/snapshot-201707281255?
wait_for_completion=true"

# Check if the backup was successful:
$ ls -al /db-backup/elasticsearch/*

# Example of a successful Elasticsearch database backup:
-rw-r--r--.  1 elasticsearch elasticsearch   39 Jul 28 10:58 /db-backup/elasticsearch/index
-rw-r--r--.  1 elasticsearch elasticsearch 4273 Jul 28 10:55 /db-backup/elasticsearch/meta-
snapshot-201707281255.dat
-rw-r--r--.  1 elasticsearch elasticsearch  907 Jul 28 10:58 /db-backup/elasticsearch/snap-
snapshot-201707281255.dat

/db-backup/elasticsearch/indices:
total 8
drwxr-xr-x. 41 elasticsearch elasticsearch 4096 Jul 28 10:55 .
drwxr-xr-x.  3 elasticsearch elasticsearch 4096 Jul 28 10:58 ..
drwxr-xr-x.  3 elasticsearch elasticsearch   51 Jul 28 10:57 audit_v2
drwxr-xr-x.  3 elasticsearch elasticsearch   51 Jul 28 10:58 documents_v2
drwxr-xr-x.  3 elasticsearch elasticsearch   51 Jul 28 10:58 draft-entities_v4
drwxr-xr-x.  3 elasticsearch elasticsearch   51 Jul 28 10:56 extracts_v2
drwxr-xr-x.  3 elasticsearch elasticsearch   51 Jul 28 10:58 -kibana
drwxr-xr-x.  3 elasticsearch elasticsearch   51 Jul 28 10:57 logstash-2017.06.23
drwxr-xr-x.  3 elasticsearch elasticsearch   51 Jul 28 10:58 logstash-2017.06.29
...
drwxr-xr-x.  3 elasticsearch elasticsearch   51 Jul 28 10:58 logstash-2017.07.27
drwxr-xr-x.  3 elasticsearch elasticsearch   51 Jul 28 10:56 logstash-2017.07.28
drwxr-xr-x.  3 elasticsearch elasticsearch   51 Jul 28 10:57 .meta_v2
drwxr-xr-x.  3 elasticsearch elasticsearch   51 Jul 28 10:58 .scripts
drwxr-xr-x.  3 elasticsearch elasticsearch   51 Jul 28 10:58 stix_v4
```

- To restore a backed up snapshot, run the following command(s):

```
$ curl -X POST 'http://localhost:9200/_snapshot/backup/snapshot-201707281255/_restore'
```

- To monitor the snapshot restore process, run the following command(s):

```
$ curl -X GET 'http://localhost:9200/_recovery/'
```

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.

- Before starting backing up data, stop Neo4j.
  Run the following command(s):

```
$ systemctl stop neo4j
```

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

```
$ systemctl status neo4j
```

- 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 -cvzf 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.conf* **configuration file** (http://neo4j.com/docs/stable/server-configuration.html) as follows:

```
# Name of the active database to mount
dbms.active_database=graph.db

# Path to the data dir containing graph.db
dbms.directories.data=${data_dir}  # ex.: neo4j/data
```

Stop the core services

- Stop systemd-managed services:
  - *postfix*
  - *redis*
  - *statsd*
  - *kibana*
  - *neo4j*
  - *elasticsearch*
  - *postgresql-10*

```
$ systemctl stop postfix redis statsd kibana neo4j elasticsearch postgresql-10
```

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)
- **Back up and restore PostgreSQL data dump with `pg_dumpall`** (https://www.postgresql.org/docs/current/static/backup-dump.html#backup-dump-all)
- **Restore PostgreSQL data with `pg_restore`** (https://www.postgresql.org/docs/current/static/app-pgrestore.html)
- **Back up PostgreSQL data with `pg_dump` and restore it with `pg_restore`** (http://postgresguide.com/utilities/backup-restore.html)
- **Restore PostgreSQL data with a continuous archive backup** (https://www.postgresql.org/docs/current/static/continuous-archiving.html#backup-pitr-recovery)
- **Restore Elasticsearch data with snapshot and restore** (https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-snapshots.html)
- **Restore a Neo4j database backup** (https://neo4j.com/docs/operations-manual/current/backup/#backup-restoring)

Reindex Elasticsearch

Before you start (re)indexing or migrating Elasticsearch, do the following:

- Make sure that ingestion, indexing, and core platform processes are *not running*.
  If they are running, stop them:

```
$ supervisorctl stop intel-ingestion:* intel-search-indexer platform-api
```

- Specify the platform settings environment variable by exporting it.
- Run `eiq-platform search reindex` to index or to reindex the Elasticsearch database.
- `search reindex` takes one argument: `index-name`.
  Its value is the name of an existing Elasticsearch index.
  Default Elasticsearch indexes for the platform:
  - *audit*
  - *documents*
  - *draft-entities*
  - *extracts*
  - *logstash-**
  - *statsd-**
  - *stix*

`search reindex` copies data from the PostgreSQL database to the Elasticsearch database, which is then indexed.

```
$ sudo -u eclecticiq -i EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/eiq-platform search reindex --index=<name_of_the_index>
```

Migrate Elasticsearch indices

To make sure you are applying the latest Elasticsearch schema, migrate Elasticsearch indices. The migration process is idempotent. It sets up and builds the required/specified indices with aliases and mappings, and it updates the index mapping templates, if necessary.

- Make sure that ingestion, indexing, and core platform processes are *not running*.
  If they are running, stop them:

```
supervisorctl stop intel-ingestion:* intel-search-indexer platform-api
```

- Specify the platform settings environment variable by exporting it.
- Run `eiq-platform search upgrade` to migrate Elasticsearch indices.
  The command runs in the background. In case of an SSH disconnection, the process should keep running normally.

This upgrade action is idempotent. First, it tries to update the Elasticsearch index mappings in-place. If it is not possible, it proceeds to reindex the existing Elasticsearch indexes.

```
$ sudo -u eclecticiq -i EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/eiq-platform search upgrade
```

Index migration log messages, if any, are printed to the terminal. If no messages are printed to the terminal, the process completed successfully.

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.

Whitelist URLs

The platform needs to access external data sources to ingest intel, as well as to enrich entities and observables. You may want to whitelist these URLs, domains and addresses, so that the platform can communicate with the external intel and service providers.

Repositories

When installing or upgrading the platform and its dependencies, the system needs to access the following source repositories.

Repository URL	Belongs to	Repo type
`https://downloads.eclecticiq.com/`	EclecticIQ Platform	deb, rpm
`https://dl.fedoraproject.org/pub/epel/7/x86_64/`	EPEL	rpm

Enrichers and feeds

Feeds and enrichers access data sources through these URLs. Whitelist the domains and allow traffic to and from them.

Domain	Belongs to	Type
`http://${elasticsearch_instance_url}:9200/${schema_resource}`	Elasticsearch sightings	enricher
`https://cybercrime-portal.fox-it.com/`	Fox-IT InTELL Portal	enricher, incoming feed
`https://api.intel471.com/v1/`	Intel 471	enricher, incoming feed
`http://api.openresolve.com/{}/{}`	OpenDNS OpenResolve	enricher
`http://${pydat_instance_url}:8000/`	PyDat	enricher
`https://stat.ripe.net/data/geoloc/{}`	RIPEstat GeoIP	enricher
`https://stat.ripe.net/data/whois/{}`	RIPEstat Whois	enricher
`https://panacea.threatgrid.com/api/v3/`	Cisco Threat Grid	enricher, incoming feed
`https://www.virustotal.com/vtapi/v2/{}`	VirusTotal	enricher
`https://endlesstunnel.info/v3`	Flashpoint AggregINT	enricher
`https://endlesstunnel.info/v3`	Flashpoint Blueprint	enricher

Domain	Belongs to	Type
https://endlesstunnel.info/v3	Flashpoint Thresher	enricher
https://api.passivetotal.org/v2	PassiveTotal Whois	enricher
https://api.passivetotal.org/v2	PassiveTotal Passive DNS	enricher
https://api.passivetotal.org/v2	PassiveTotal IP/Domain	enricher
https://api.passivetotal.org/v2	PassiveTotal Malware	enricher
http://${splunk_instance_url}:8089/	Splunk sightings	enricher
http://api.domaintools.com/v1/{}/host-domains	DomainTools Hosted Domains	enricher
http://api.domaintools.com/v1/reputation	DomainTools Reputation	enricher
https://api.domaintools.com/v1/{}/host-domains	DomainTools Suspicious Domains	enricher
https://api.isightpartners.com/search/{}	FireEye iSIGHT	enricher
https://api.recordedfuture.com/live/sc/entity/{}	Recorded Future	enricher
https://unshorten.me/s/{}	Unshorten-URL	enricher
https://api.dnsdb.info/{}	Farsight DNSDB	enricher
https://www.threatcrowd.org/{}	ThreatCrowd	enricher
https://censys.io/api/v1/search/ipv4	Censys	enricher
http://api.domaintools.com/v1/{}/name-server-domains/	DomainTools Malicious Server Domains	enricher
http://api.domaintools.com/v1/{}/whois/parsed	DomainTools Parsed Whois	enricher
https://intelapi.crowdstrike.com/indicator/v1/search/{}	Crowdstrike Falcon Intelligence Indicator	enricher
http://api.domaintools.com/v1/reverse-whois/{}	DomainTools Reverse Whois	enricher
https://cve.circl.lu/api/cve/	CVE Search	enricher
https://www.circl.lu/v2pssl/cquery/{}	CIRCL IPs related to SSL certificate	enricher
https://www.circl.lu/v2pssl/cfetch/{}	CIRCL SSL Certificate Fetcher	enricher
https://api.shodan.io/shodan/	Shodan	enricher
https://api.spycloud.io/sp-v1/breach	SpyCloud Breach Data	enricher
https://api.emaildefense.proofpoint.com/v1	Proofpoint Email Threat	enricher

Domain	Belongs to	Type
`http://${misp_instance_url}/`	MISP API	enricher
`/absolute/path/to/GeoLite2-City.mmdb`	MaxMind GeoIP	enricher
`https://nti.nsfocusglobal.com/api/v1/search/{}`	NSFocus Intelligence	enricher

Other URLs

Domain	Belongs to	Type
`http://${variable_subdomain}.cyberfeed.net:${port_number}`	AnubisNetworks	incoming feed
`https://www.threathq.com/`	PhishMe Intelligence	incoming feed
`https://api.threatrecon.co/`	Threat Recon	incoming feed
`http://hailataxii.com`	Hail a TAXII	open source cyber threat intelligence source
`https://test.taxiistand.com/`	TAXII Stand	public OpenTAXII test server

Open ports

The platform components communicate with the platform and with each other through these ports. Make sure they are open within the platform network.

Port	Belongs to
`9200`	elasticsearch
`5601`	kibana
`7474; 7473`	neo4j
`4008`	neo4j-batching
`8008`	platform-api
`5432`	postgresql-10
`6379`	redis
`6755`	logstash
`80; 443`	nginx

Port	Belongs to
`25; 587`	postfix
`9001`	supervisor
`8125`	statsd

Config and log files

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

Configuration, log and manifest files

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 template config files to use as boilerplates or scaffolding
$ 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

# Supervisor config file
/etc/supervisord.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

# Nginx web server config file
/etc/nginx/nginx.conf
/opt/eclecticiq/etc/nginx/conf.d/platform.conf

# Postfix email server config file
/etc/postfix/main.cf

# StasD config file
/etc/statsd/config.js # CentOS, RHEL
/opt/statsd/config.js # Ubuntu

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

# Elasticsearch config files
/etc/elasticsearch/elasticsearch.yml
/etc/elasticsearch/logging.yml

# 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 # CentOS, RHEL
/opt/eclecticiq/etc/default/logstash # Ubuntu

# Kibana config file
/etc/kibana/kibana.yml

# Redis message broker config file
/etc/redis.conf # CentOS, RHEL
/etc/redis/redis.conf # Ubuntu

# Neo4j config file
/etc/neo4j/neo4j.conf
```

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 key value, autl URLs ponting to external components Celery-managed tas
`/opt/eclecticiq/etc/eclecticiq/opentaxii.yml`	Contains **OpenTAXII** (`https://opentaxii.readthedocs` URL and port for the service, as well as the designated inb
`/opt/eclecticiq/etc/eclecticiq/proxy_url`	Contains the IP addresses and host names that should byp comma-separated. The no-proxy list needs to always inclu `127.0.0.1,localhost`.
`/opt/eclecticiq/etc/kibana-dashboard.json`	Contains the configuration defining the Kibana dashboard l web-based GUI.
`/opt/eclecticiq/etc/logstash/conf.d/platform-api.conf`	Defines the locations of the log files storing log data relatec API, intel ingestion, and tasks.
`/opt/eclecticiq/etc/logstash/conf.d/platform-ui.conf`	Defines the locations of the log files storing log data relatec Nginx is the web server; if it is not working normally or if it i may be unavailable.
`/opt/eclecticiq/etc/logstash/conf.d/input.conf`	Elasticsearch Curator input configuration file. **Elasticsearc** (`https://www.elastic.co/guide/en/elasticsearch/c` manages Elasticsearch indices.
`/opt/eclecticiq/etc/logstash/conf.d/output.conf`	Defines the Elasticsearch cluster and the data transfer prot to Elasticsearch.
`/opt/eclecticiq/etc/logstash/conf.d/filters.conf`	Defines filters as needed to select specific indices or data t
`/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 log data relatec Nginx is the web server OpenTAXII relies on.
`/opt/eclecticiq/etc/nginx/conf.d/platform.conf`	Defines the platform configuration the Nginx web server ne key, ports, endpoints to make available services like the TA documentation, and so on.
`/opt/eclecticiq/etc/sysconfig/logstash`	INI configuration file defining the Logstash heap size. Defa GB). Location on CentOS and RHEL OSs: */opt/eclecticiq/e* */opt/eclecticiq/etc/default/logstash*.
`/opt/eclecticiq/etc/supervisord.d/platform-api.ini`	Supervisor INI configuration file to start the platform core s
`/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 c
`/opt/eclecticiq/etc/supervisord.d/search-ingestion.ini`	Supervisor INI configuration file to start the Elasticsearch ir
`/opt/eclecticiq/etc/supervisord.d/neo4j-batching.ini`	Initializes the `neo4j-batching` process. By default, the pro

File name and location	Description
/opt/eclecticiq/etc/supervisord.d/opentaxii.ini	Supervisor INI configuration file to start OpenTAXII as the a incoming and outgoing feeds.
/opt/eclecticiq/etc/supervisord.d/task-workers.ini	Supervisor INI configuration file to start the Celery-manage integrations, enrichers, utilities, and beat.
/etc/kibana/kibana.yml	Kibana configuration file. Check it and edit or update it, if n newer version.
/etc/ (CentOS/RHEL); /etc/redis/ (Ubuntu);redis.conf	Defines the configuration for the Redis message broker. It dataset snapshot autosave time intervals, and so on.
/etc/elasticsearch/elasticsearch.yml	Elasticsearch configuration file. Check it and edit or update **enough memory to the heap size** (https://www.elastic.co/guide/en/elasticsearch/g based on your system configuration and the size of the Ela recommended value is 4 GB (default value) : ES_HEAP_SIZ
/etc/elasticsearch/logging.yml	Configures logging and sets the logging level (info, warning
/etc/neo4j/neo4j.conf	Defines Neo4j set up and configuration such as maximum database dir and port, and Java settings for Neo4j such as for the graph database. Default min. and max. values: 409
/etc/nginx/nginx.conf	Defines the configuration options for the Nginx web server.
/etc/rc.d/init.d/kibana	Script initializing the /usr/share/kibana/bin/kibana pro By default, the process is configured to start automatically.
/etc/supervisord.conf	Contains the **Supervisor configuration** (http://superv supervisord and supervisorctl. This file should include (http://supervisord.org/configuration.html#inet- enabling the supervisord server to reply to system health
/media/pgsql/10/data/ (CentOS, RHEL); /etc/postgresql/10/main/ (Ubuntu);postgresql.conf	**PostgreSQL configuration file** (https://www.postgres setting.html).
/etc/postfix/main.cf	**Postfix configuration file** (http://www.postfix.org/d configuring email addreses through the GUI, you need to d host and SMTP in this file.
/opt/eclecticiq/etc/statsd/ (CentOS, RHEL); /opt/statsd/ (Ubuntu);config.js	**StatsD configuration file** (https://github.com/etsy/s
/opt/eclecticiq/etc/nginx/conf.d/platform.conf	Enables Nginx to pick up the platform configuration and to

Log files

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

```
$ 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.
`/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.

File name and location	Description
`/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-enricher*.log`	They list enricher tasks, their intel providers, enricher priorities, and enricher-related utility tasks running in the background.
`/var/log/eclecticiq/task-worker-incoming*.log`	They list integrations such as incoming feeds, their intel providers, incoming feed priorities, and incoming feed-related tasks running in the background.
`/var/log/eclecticiq/task-worker-outgoing*.log`	They list integrations such as outgoing feeds, their intel providers, outgoing feed priorities, and outgoing feed-related 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.
`/var/log/logstash/logstash-plain.log`	Most recent Logstash events log file.
`/var/log/logstash/logstash-plain-YYYY-MM-DD.log`	Historical Logstash events log files.
`/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.

File name and location	Description
`/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.
`/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
```

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.

Default users

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

The installation procedure creates several default user profiles at platform level, as well as at host 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 on CentOS\RHEL. Web server user and group on Ubuntu is `www-data`.	`/var/cache/nginx`
`postgres`	`postgres`	✖	PostgreSQL database	Database user, can access the default `platform` database	`/var/lib/pgsql`
`redis`	`redis`	✖	Redis server, message broker and queue manager	Redis database and message broker user	`/var/lib/redis`
`kibana`	`kibana`	✖	Kibana	Kibana, a data visualization plugin for Elasticsearch	`/opt/kibana/bin/kibana`

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
$ sudo su -

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

# run this command to login as root with elasticsearch user
$ su -s /bin/bash elasticsearch

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

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

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

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

# run this command to login as root with redis user
$ 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.