Install and configure EclecticIQ Platform

Installation and configuration on Ubuntu Server — For system administrators

Last generated: October 20, 2017

Table of contents

Ubuntu installation and configuration guide

This document guides you through the installation, setup and configuration of EclecticIQ Platform when you choose to install the product from a DEB package on a target system running Ubuntu Server 16.04.

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 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

Last generated on Oct 20, 2017

Before you start

Review the system requirements before proceeding to install the platform on Ubuntu from a DEB package.

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

HW area	Minimum	Recommended	Notes
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
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

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

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

Operating systems

Supported operating systems:

- **Ubuntu Server 16.04 LTS (Xenial Xerus)** `(https://wiki.ubuntu.com/xenialxerus/releasenotes)`

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 use this 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.

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 download page** (http://www.oracle.com/technetwork/java/javase/downloads/index.html)
PostgreSQL	9.5	**PostgreSQL documentation about installing on Ubuntu** (https://www.postgresql.org/download/linux/ubuntu/)
Redis	3.2.3	**Redis download page** (http://redis.io/download)
Nginx	1.12	**Nginx download page** (https://nginx.org/download/)
Neo4j	2.3.8 Community	**Neo4j download page** (http://neo4j.com/download/)
Elasticsearch	2.4.4	**Elasticsearch installation documentation** (https://www.elastic.co/guide/en/elasticsearch/reference/master/deb.html)
delete-by-query	n/a	**delete-by-query installation documentation** (https://www.elastic.co/guide/en/elasticsearch/plugins/2.3/plugins-delete-by-query.html)
elasticdump	2.4.2	**Install with `npm` as a Node js module** (https://www.npmjs.com/package/elasticdump)
Logstash	2.4.1	**Logstash install instructions** (https://www.elastic.co/guide/en/logstash/2.3/installing-logstash.html)
Kibana	4.5.1	**Kibana install instructions** (https://www.elastic.co/guide/en/kibana/4.5/setup-repositories.html)
unzip	n/a	**Install with `apt-get install zip unzip` on Ubuntu Server** (https://www.digitalocean.com/community/questions/how-to-install-zip-in-ubuntu)

Dependency	Version	Reference
Node.js	6.x or later	**Node.js for Ubuntu Server (includes *npm*)** (https://nodejs.org/en/download/package-manager/#debian-and-ubuntu-based-linux-distributions)
poppler-utils	n/a	**Install with `apt-get install poppler-utils` on Ubuntu Server** (https://packages.ubuntu.com/xenial/python/poppler-utils)
Supervisor	n/a	**Install with `apt-get install supervisor` on Ubuntu Server** (https://www.howtoinstall.co/en/ubuntu/xenial/supervisor)
Postfix	n/a	**Install with `apt-get install postfix` on Ubuntu Server** (https://help.ubuntu.com/lts/serverguide/postfix.html)
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.

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.

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

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/v2/`	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
`https://endlesstunnel.info/v3`	Flashpoint Thresher	enricher

Domain	Belongs to	Type
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://app.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

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 intillegence 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-9.5
`6379`	redis
`6755`	logstash
`80; 443`	nginx
`25; 587`	postfix
`9001`	supervisor
`8125`	statsd

Install the dependencies

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

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 - {user_name}
```

Or:

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

```
$ sudo {command}
```

Create the APT repository for the dependencies

You can download the core dependencies for the platform from a centralized APT 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 APT repository provides the following platform dependencies:

- Nginx
- Redis-server
- PostgreSQL
- Node.js, including npm
- Elasticsearch, including plugins
- Logstash
- Kibana
- Neo4j
- StatsD
- Supervisor
- Postfix

Add GPG key and APT repository

- Retrieve the GPG key by running the `gpg` command twice *as root*, as shown below:

```
# Switch to root user
$ sudo -i

# Run gpg twice
$ gpg --recv-keys 379CE192D401AB61
$ gpg --recv-keys 379CE192D401AB61

# Successful response 1
...
gpg: Total number processed: 1
gpg:               imported: 1   (RSA: 1)

# Successful response 2
...
gpg: Total number processed: 1
gpg:              unchanged: 1
```

- Add the GPG key by running the following command(s):

```
$ gpg --export 379CE192D401AB61 | apt-key add

# Successful response
OK
```

Warning:
If the command returns an error message like the following one:

```
The following signatures couldn't be verified because the public key is not available:
NO_PUBKEY 379CE192D401AB61
```

run the previous command again:

```
$ gpg --export 379CE192D401AB61 | apt-key add
```

The command response should be `OK`:

```
# Successful response
OK
```

- Add the APT repository to the list of allowed sources by replacing the **{user_name}** and **{password}** placeholders in the URL with *your EclecticIQ login credentials*:

```
$ echo 'deb https://{user_name}:{password}@downloads.eclecticiq.com/platform-deb-deps xenial 2.0'
> /etc/apt/sources.list.d/eclecticiq-deps.list
```

- Set the repository preferences and pin it, so that you remain on a stable platform release channel:

```
$ printf "Package: *\nPin: origin downloads.eclecticiq.com\nPin-Priority: 990" >
/etc/apt/preferences.d/repo
```

- Download and update package information for all configured sources:

```
$ apt-get update
```

Install the core dependencies

- Install the following core dependencies:

```
$ apt-get install supervisor nginx postgresql-9.5 redis-server elasticsearch=2.4.4
logstash=1:2.4.1-1 kibana=4.5.1 neo4j=2.3.8 statsd
```

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

```
$ apt-mark hold elasticsearch logstash kibana neo4j
```

Install the other required dependencies

PostgreSQL

- Install these additional PostgreSQL packages by running the following command(s):

```
$ apt-get install -y postgresql-contrib-9.5 postgresql-server-dev-9.5
```

Node.js

- Install **Node.js 6.x or later** (https://nodejs.org/en/download/package-manager/#debian-and-ubuntu-based-linux-distributions) by replacing the **{user_name}** and **{password}** placeholders in the URL with *your EclecticIQ login credentials*:

```
# Remove any previous Node.js and npm installations
# to avoid dependency issues
$ apt-get remove nodejs-* npm

# Retrieve Node.js
$ curl --silent --location https://{user_name}:{password}@downloads.eclecticiq.com/platform-deb-deps/pool/n/node.js/setup_6.x | bash -

# Install Node.js
$ apt-get install -y nodejs-6.11.4
```

- Verify that Node.js is correctly installed:

```
$ node --version

# Example response:
v6.11.4
```

Elasticsearch plugins

- Install *delete-by-query* by replacing the **{user_name}** and **{password}** placeholders in the URL with *your EclecticIQ login credentials*:

```
$ /usr/share/elasticsearch/bin/plugin install
https://{user_name}:{password}@downloads.eclecticiq.com/platform-deb-deps/pool/d/delete-by-query/delete-by-query-2.4.4.zip
```

In case the standard installation fails because of a Java CA certificate error, retry the operation by by running the following command(s):

```
$ /var/lib/dpkg/info/ca-certificates-java.postinst configure

$ /usr/share/elasticsearch/bin/plugin install
https://{user_name}:{password}@downloads.eclecticiq.com/platform-deb-deps/pool/d/delete-by-query/delete-by-query-2.4.4.zip
```

Install the plugin on each node in the cluster.
After installation, restart each node for the changes to become effective.

- Install *elasticdump* 2.4.2 by replacing the **{user_name}** and **{password}** placeholders in the URL with *your EclecticIQ login credentials*:

```
# Install globally with '-g'
$ npm install -g https://{user_name}:{password}@downloads.eclecticiq.com/platform-deb-deps/pool/e/elasticdump/elasticdump-2.4.2.tgz
```

Neo4j

- Set up a profile file for Neo4j where you define the following environment variables:

```
$ echo -e 'export NEO4J_HOME=/usr/share/neo4j\nexport PATH=$PATH:$HOME/bin:$NEO4J_HOME' >
/etc/profile.d/neo4j.sh
```

- Reload the Neo4j profile to update it, so that it can pick up the newly defined environment variables:

```
$ source /etc/profile.d/neo4j.sh
```

After installing the platform, make sure these environment variables are accessible to the `neo4j` user, `nogroup` group profile.

```
$ su - neo4j -c 'echo $NEO4J_HOME; echo $PATH'
```

```
# Expected output
/usr/share/neo4j
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/snap/bin:/var/lib/neo4j/bin:/usr/share/ne
```

Postfix

- **Install** `(https://help.ubuntu.com/lts/serverguide/postfix.html)` **Postfix** `(http://www.postfix.org/)`:

```
$ DEBIAN_PRIORITY=low apt-get install postfix
```

For a thorough tutorial and walkthrough to install and configure Postfix on Ubuntu Server 16.04, see **How to install and configure Postfix on Ubuntu 16.04** `(https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-postfix-on-ubuntu-16-04)`.

The recommended default settings during the installation are suitable to get the email server jump-started. You need to fine-tune them to accurately reflect your system setup, and to make the configuration robust enough for a production environment.

Java Development Kit

Obtain **Oracle Java SE Development Kit 8** `(http://www.oracle.com/technetwork/java/javase/downloads/)` by carrying out the following steps:

- Download Oracle Java JDK 8 by replacing the **{user_name}** and **{password}** placeholders in the URL with *your EclecticIQ login credentials*:

```
$ wget -qO- https://{user_name}:{password}@downloads.eclecticiq.com/java/1.14/jdk-8u131-linux-x64.tar.gz | tar -xzf - -C /opt/
```

- The default Java installation directory is a child of the */opt* directory.
  Verify that the Java JDK package is successfully extracted to the */opt* directory:

```
$ ls /opt
```

- Install Oracle Java JDK by running the following command(s):

```
$ update-alternatives --install /usr/bin/java java /opt/jdk1.8.0_131/bin/java 100
$ update-alternatives --install /usr/bin/javac javac /opt/jdk1.8.0_131/bin/javac 100
```

- Set the default Java version for the target system, and make sure the system uses the Oracle JDK:

```
# Run this command to set the default Java
$ update-alternatives --set java /opt/jdk1.8.0_131/bin/java
```

- Verify the JDK version installed on the target system:

```
$ java -version

# Expected output
java version "1.8.0_131"
Java(TM) SE Runtime Environment (build 1.8.0_131-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)
```

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

```
$ echo 'export JAVA_HOME=/opt/jdk1.8.0_131' > /etc/profile.d/jdk.sh
```

- Reload the JDK profile to update it, so that it can pick up the newly defined environment variable:

```
$ source /etc/profile.d/jdk.sh
```

- Check the `JAVA_HOME` environment variable by running the following command(s):

```
$ echo $JAVA_HOME

# Expected output
/opt/jdk1.8.0_131
```

Python packages

- Install the following Python packages, as they are required platform dependencies:

```
$ apt-get install -y python python3 python-dev python3-dev libxml2-dev python-libxslt1 libyaml-dev python-yaml python-psycopg2
```

poppler-utils

The platform requires **poppler-utils** (https://poppler.freedesktop.org/) to **process PDF** (https://en.wikipedia.org/wiki/poppler_(software)#poppler-utils) files.

- Install ***poppler-utils*** (https://packages.ubuntu.com/xenial/python/poppler-utils) by running the following command(s):

```
$ apt-get install -y poppler-utils
```

Install via a DEB package

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

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 - {user_name}
```

Or:

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

```
$ sudo {command}
```

Create the APT repository for the platform

Before beginning the installation procedure, make sure you have the necessary login credentials to access the provided resource download locations.

Add GPG key and APT repository

- Download the PGP public key and add it to the list by running the following command(s):

```
$ wget -qO- https://downloads.eclecticiq.com/public/GPG-KEY-eclecticiq | apt-key add -

# Successful response
OK
```

- Add the EclecticIQ Platform repository to the list of configured sources by running the following command(s):

```
$ echo 'deb https://{user_name}:{password}@downloads.eclecticiq.com/eiq-platform-deb xenial
2.0.1' > /etc/apt/sources.list.d/eclecticiq-platform.list
```

- Download and update package information for all configured sources:

```
$ apt-get update
```

Install the platform

- Install EclecticIQ Platform 2.0 by running the following command(s):

```
$ apt-get install -y eclecticiq-platform
```

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. For example, to update to a more recent version of the help.

To install the platform documentation on *Ubuntu Server*, so that it is available in-product as a help resource, do the following:

```
# {version_number} format: 0.0.0-0
$ apt-get install -y eclecticiq-platform-docs-{version_number}

# Install a specific version of the documentation
# by specifying the release number:
$ apt-get install -y eclecticiq-platform-docs-2.0.1-1

# Install the latest version of the documentation
$ apt-get install -y eclecticiq-platform-docs
```

The default install location for the platform documentation is */opt/eclecticiq/platform/docs*.

Check the version

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

It is possible to upgrade the platform only sequentially, it is not possible to skip versions.
For example, if you want to upgrade from release 1.10.0 to 1.14.1, you need to upgrade step by step by installing all intermediate releases until 1.14.1 included.

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

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

exit 99
```

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

Check permissions

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

```
# Create the dir if it does not exist:
$ mkdir -p /var/run/supervisor/

# Set ownership:
$ chown -R eclecticiq:eclecticiq /var/run/supervisor/
```

```
$ chown -R eclecticiq:eclecticiq /etc/supervisor/
```

```
$ 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
```

Enable Supervisor to automatically start at system boot:

```
$ systemctl enable supervisor
```

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.

Last generated on Oct 20, 2017

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

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

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

# Logstash heap size
/opt/eclecticiq/etc/sysconfig/logstash # CentOS, RHEL
/opt/eclecticiq/etc/default/logstash # Ubuntu

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

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

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

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

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

# Nginx web server config file
/etc/nginx/nginx.conf
/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
```

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

File name and location	Description

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`.
`/opt/kibana/config/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.properties`	Defines the maximum size for page cache and log file
`/etc/neo4j/neo4j-server.properties`	Defines the configuration options of the Neo4j graph datab communication port, and so on.
`/etc/neo4j/neo4j-wrapper.conf`	Defines the Java settings related to Neo4j like minimum an graph database. Default min. and max. values: 4096 MB.
`/etc/nginx/nginx.conf`	Defines the configuration options for the Nginx web server.
`/opt/kibana/bin/kibana`	Script initializing the `/opt/kibana/bin/kibana` process as 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
`/var/lib/pgsql/9.5/data/ (CentOS, RHEL); /etc/postgresql/9.5/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.
`/etc/statsd/ (CentOS, RHEL); /opt/statsd/ (Ubuntu);config.js`	**StatsD configuration file** `(https://github.com/etsy/s`
`/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.

File name and location	Description
`/var/log/eclecticiq/search-ingestion.log`	Search indexing log file. It logs information on Elasticsearch data indexing.
`/var/log/eclecticiq/kibana.log`	It logs information on Kibana dashboard like connection and availability.
`/var/log/eclecticiq/neo4j.log`	Neo4j graph database log file. It logs information on Neo4j server and database operation.
`/var/log/eclecticiq/opentaxii.log`	It logs OpenTAXII server log information.
`/var/log/eclecticiq/task-beat.log`	It logs heartbeat timestamp information. The heartbeat interval is 30 seconds.
`/var/log/eclecticiq/task-worker-reindexing.log`	It lists synced enricher tasks, intel providers, feed transport types, and platform utility tasks. Indexing and syncing go through Redis and Celery.
`/var/log/eclecticiq/task-worker-enrichers.log`	It lists enricher tasks, intel providers, feed transport types, and platform utility tasks running in the background.
`/var/log/eclecticiq/task-worker-integrations.log`	It lists integrations like enricher tasks, intel providers, and platform utility tasks running in the background.
`/var/log/eclecticiq/task-worker-providers.log`	It lists integrations like enricher tasks, intel providers, and platform utility tasks running in the background.
`/var/log/eclecticiq/task-worker-utilities.log`	It lists integrations like enricher tasks, intel providers, and platform utility tasks running in the background.
`/var/log/logstash/logstash.err`	It logs Logstash errors and error messages.
`/var/log/logstash/logstash.log`	It logs Logstash events.
`/var/log/logstash/logstash.stdout`	Standard output format for Logstash log information.
`/var/log/elasticsearch/intel.log`	Elasticsearch log file. It logs Elasticsearch events like initialization, startup, designated Elasticsearch cluster, and so on.
`/var/log/elasticsearch/intel.log.YYYY-MM-DD.log`	Elasticsearch log file for a specific date. *YYYY-MM-DD* (year, month, day) in the file name is replaced by the date the log information refers to. It logs Elasticsearch events like initialization, startup, designated Elasticsearch cluster, and so on.
`/var/log/elasticsearch/intel_index_indexing_slowlog.log`	Elasticsearch log file. It logs Elasticsearch indexing information.
`/var/log/elasticsearch/intel_index_search_slowlog.log`	Elasticsearch log file. It logs Elasticsearch search index information.

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`	`nogroup`	✖	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:

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

Configure the platform

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

After installing the platform, you can proceed to configuring it, along with the required third-party components such as web server, graph, indexing, and search:

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

Obtain root access

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 - {user_name}
```

Or:

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

```
$ sudo {command}
```

Reload Supervisor after configuration 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 supervisor

$ supervisorctl reload
```

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

Reload systemd units after configuration changes

After editing *systemd* unit (*.service*) or init (*.ini*) files, reload the systemd manager configuration to update unit/init configuration files, and to apply any changes to the *.service* or to the *.ini* files:

```
$ systemctl daemon-reload
```

This applies to all *systemd*-managed programs, applications, and services in the platform.

Configure PostgreSQL

Initialize the database

- Switch to the `eclecticiq` user to initialize PostgreSQL and to set the target directory to store the database to:

```
$ su - eclecticiq -c "/usr/lib/postgresql/9.5/bin/initdb PGDATA"
```

`PGDATA` is an environment variable PostgreSQL creates during installation.
By default, `PGDATA` points to */var/lib/postgresql/9.5/main*.

Enable the database service

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

```
$ systemctl enable postgresql@9.5-main
```

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

```
$ systemctl start postgresql@9.5-main
```

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

```
$ systemctl status postgresql@9.5-main
```

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** (http://www.postgresql.org/docs/current/static/tutorial-createdb.html) a new database (in the example: `platform`):

```
$ su - postgres -c "psql -c \"CREATE DATABASE platform 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 (in the example: `test`/`test`, respectively):

```
$ su - postgres -c "psql -c \"CREATE USER test WITH PASSWORD 'test';\""
```

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

```
$ su - postgres -c "psql -c \"GRANT ALL PRIVILEGES ON DATABASE platform TO test;\""
```

Set the database authentication method

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

```
$ nano /etc/postgresql/9.5/main/pg_hba.conf
```

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

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

Edit the database configuration

Make sure that the `data_directory` and `hba_file` properties in the *postgresql.conf* configuration point to the actual locations of the main database and the database host-based authentication configuration, respectively:

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

```
$ nano /etc/postgresql/9.5/main/postgresql.conf
```

- Verify that the following paths match the actual file locations in your system:

```
data_directory = "/var/lib/postgresql/9.5/main"          # The main database lives here
hba_file       = "/etc/postgresql/9.5/main/pg_hba.conf"  # Host-based authentication config
```

- Depending on your system setup and on the total amount of memory you can allocate to PostgreSQL, you may wish to **tune** `(http://pgtune.leopard.in.ua/)` the database configuration.
  The following extract is just an example for reference, it is not a **silver bullet** `(https://en.wikipedia.org/wiki/no_silver_bullet)`:

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

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

# Maintenance operations are usually not performed concurrently.
# This means that they can use significantly more memory to run faster.
maintenance_work_mem = 1GB
```

Restart the database service

- Restart the PostgreSQL service:

```
$ systemctl restart postgresql@9.5-main
```

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

```
$ systemctl status postgresql@9.5-main
```

- Verify that user and database are successfully created:

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

Add the database to the platform settings

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

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

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

`username`	Replace this placeholder with the user name of the user that can access the database. Example: `test`

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

Configure Nginx

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

To configure Nginx, do the following:

- Back up *nginx.conf*:

```
$ mv /etc/nginx/nginx.conf /etc/nginx/nginx.conf.backup
```

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

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

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

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

Set Nginx user and group

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

- The `user` should have the default Nginx values set to `www-data` for both user and group name:

```
user    www-data www-data;
```

Check access log and log format

- Verify the Nginx **access log** `(https://www.nginx.com/resources/admin-guide/logging-and-monitoring/)` format to make sure Nginx can recognize and consume the *expected format for web server logs*.
  The *access.log* file location is */var/log/nginx/*.
- In the *nginx.conf* file, make sure that the `log_format` directive holds the configuration parameters for the web server **log format** `(https://nginx.org/en/docs/http/ngx_http_log_module.html#log_format)`.
  The *nginx.conf* file location is */etc/nginx/*.
  The following example shows the default `log_format` section in the Nginx *nginx.conf* file, as per Nginx 1.8:

```
# Recommended basic Nginx configuration from EclecticIQ

user       www-data www-data;
error_log  /var/log/nginx/error.log info;
pid        /run/nginx.pid;
worker_processes auto;

events {
  worker_connections 1024;
}

http {
  server_tokens off;
  include mime.types;
  default_type application/octet-stream;

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

  access_log /var/log/nginx/access.log json;
  include /etc/nginx/conf.d/*.conf;
}
```

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

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

- To enable Nginx to pick up and start the platform configuration, copy 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:

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

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

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

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

- Make sure that the **SSL certificate paths** `(https://nginx.org/en/docs/http/configuring_https_servers.html)` point to the correct locations where your certificates are stored.

  Example:

```
ssl_certificate      /etc/ssl/certs/<certificate_name>.crt;
ssl_certificate_key  /etc/ssl/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/ssl/private/platform_local.key
-out /etc/ssl/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 Redis

To configure Redis, do the following:

- Back up *redis.conf*:

```
$ mv /etc/redis/redis.conf /etc/redis/redis.backup
```

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

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

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

Check the service configuration

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

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.

> **Warning:**
> 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

- Run the following command to add a directive to the Redis configuration file that systemd uses to manage the service:

```
$ sed -i -r 's|^(\[Service\])\s*$|\1\nReadWriteDirectories=/media/redis|g'
/etc/systemd/system/redis.service
```

It grants Redis read/write access to the `dir` location storing the snapshot database defined in *redis.conf*.

After editing *systemd* unit (*.service*) or init (*.ini*) files, reload the systemd manager configuration to update unit/init configuration files, and to apply any changes to the *.service* or to the *.ini* files:

```
$ systemctl daemon-reload
```

This applies to all *systemd*-managed programs, applications, and services in the platform.

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:

```
$ mkdir -p /media/redis
```

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

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

```
$ chown -R redis:redis /var/lib/redis
```

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

Check the platform settings

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

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

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

```
$ redis-server --version

# Expected output
Redis server v=3.0.6 sha=00000000:0 malloc=jemalloc-3.6.0 bits=64 build=687a2a319020fa42

$ redis-cli --version

# Expected output
redis-cli 3.0.6
```

Enable the service

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

```
$ systemctl enable redis-server
```

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

```
$ systemctl start redis-server
```

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

```
$ systemctl status redis-server
```

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

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

# Redis responds
PONG
```

Configure Elasticsearch

To configure Elasticsearch, do the following:

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

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

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

- Set `path.data` to the location where Elasticsearch stores its data.
  Default location: */media/elasticsearch*

Disable dynamic scripting

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

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

Example *elasticsearch.yml* file for reference:

```
cluster.name: intel

discovery.zen.ping.multicast: false

index.number_of_replicas: 0

index.number_of_shards: 1

node.local: true

path.data: /media/elasticsearch

script.indexed: true

script.inline: false
```

Set heap size and custom tmp dir

You should allocate enough memory to the **heap size** (`https://www.elastic.co/guide/en/elasticsearch/guide/current/heap-sizing.html`), based on your system configuration and the size of the Elasticsearch index.
The minimum recommended value is 4 GB (default value): `ES_HEAP_SIZE=4g`.

You can set this option in the Elasticsearch configuration file:

- Go to */etc/default* and open the *elasticsearch* configuration file:

```
$ nano /etc/default/elasticsearch
```

Example *elasticsearch* file for reference:

```
ES_HOME=/usr/share/elasticsearch
ES_HEAP_SIZE=4g
ES_JAVA_OPTS="-Djna.tmpdir=/tmp/elasticsearch"
MAX_OPEN_FILES=65535
MAX_MAP_COUNT=262144
LOG_DIR=/var/log/elasticsearch
DATA_DIR=/media/elasticsearch
WORK_DIR=/tmp/elasticsearch
CONF_DIR=/etc/elasticsearch
ES_USER=elasticsearch
```

- Make sure you set the Elasticsearch heap size to at least 4 GB or more, if your system allows it:

```
ES_HEAP_SIZE=4g
```

- Optionally, you may **change the default temp directory** (https://github.com/elastic/elasticsearch/issues/18406) and set a custom temporary directory, if the current one is mounted with the `noexec` option.
  To do so, **edit or add** (https://www.elastic.co/guide/en/elasticsearch/reference/2.4/setup-configuration.html#setup-configuration-memory) the `ES_JAVA_OPTS` property:

```
ES_JAVA_OPTS="-Djna.tmpdir=/tmp/elasticsearch"
```

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:

```
$ mkdir -p /media/elasticsearch
```

- Make sure the correct user owns the target directory to access it in read and write modes.

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

- Verify that the `elasticsearch` user can read and write to the temp directory.
  If necessary, set it by running the following command(s):

```
$ chown -R elasticsearch:elasticsearch /tmp/elasticsearch
```

Check the Elasticsearch URL

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

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

Enable the service

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

```
$ systemctl enable elasticsearch
```

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

```
$ systemctl start elasticsearch
```

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

```
$ systemctl status elasticsearch
```

If you install additional plugins after configuring Elasticsearch, always restart the Elasticsearch service after completing the plugin installation:

```
$ systemctl restart elasticsearch
```

Configure Logstash

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

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

These are the Logstash configuration files related to the platform:

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

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

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

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

Enable the service

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

```
$ systemctl enable logstash
```

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

```
$ systemctl start logstash
```

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

```
$ systemctl status logstash
```

Configure Kibana

The Kibana configuration file is */opt/kibana/config/kibana.yml*.

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

The default parameters and values should not need any editing:

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

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.
  Typically, the IP address corresponds to the one of the host machine the Kibana instance runs on.

  The IP address and port number specified here need to match the `port` and `host` values defined in the */opt/kibana/config/kibana.yml* configuration file.

  Example:

```
KIBANA_URL = 'http://127.0.0.1:5601'
```

- Verify that `SYSTEMD_SERVICES` in */opt/eclecticiq/etc/eclecticiq/platform_settings.py* includes the `kibana` service:

```
SYSTEMD_SERVICES = [
    'elasticsearch',
    'logstash',
    'postfix',
    'neo4j-service'
    'postgresql@9.5-main'
    'redis-server'
    'statsd',
    'kibana',
]
```

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

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

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

Enable the service

To set up Kibana as a *systemd*-managed service, do the following: - Copy the *kibana.service* file shipped with the platform to */etc/systemd/system*:

```
$ cp /opt/eclecticiq/etc-extras/kibana.service /etc/systemd/system/kibana.service
```

- Set the `kibana` user and the `kibana` group as the owners of the service:

```
$ sed -i 's/eclecticiq/kibana/g' /etc/systemd/system/kibana.service
```

- Open *kibana.service*:

```
$ nano /etc/systemd/system/kibana.service
```

- The *systemd* unit file should be ready for use as is without any further editing.
  For reference:

```
[Unit]
Description=Kibana
Documentation=http://www.elastic.co
Requires=elasticsearch.service

[Service]
User=kibana
Group=kibana
ExecStart=/opt/kibana/bin/kibana
StandardOutput=journal
StandardError=journal
Restart=always

[Install]
WantedBy=multi-user.target
```

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

```
$ systemctl enable kibana
```

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

```
$ systemctl start kibana
```

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

```
$ systemctl status kibana
```

Access Kibana

- To access Kibana, in the web browser address bar enter a URL with the following format:
  https://<platform_host>/api/kibana/app/kibana#/
  Keep the trailing */*
  Example: *https://platform.host.com/api/kibana/app/kibana#/*

Configure StatsD

To configure StatsD, do the following:

- If */opt/statsd/config.js* already exist, open it and edit it to reflect your actual configuration.
  Otherwise, create it:

```
$ nano /opt/statsd/config.js
```

In the the *config.js* configuration file check the following points:

- Verify that `backends` is assigned **`statsd-elasticsearch-backend`** (`https://github.com/markkimsal/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.
- Save your changes, and close the configuration file.

Use the following example as a scaffolding template to customize, based on your system setup:

```
{
  backends:          [ 'statsd-elasticsearch-backend'],
  flushInterval:     30000,  // 30 Seconds
  deleteIdleStats:   true,

  elasticsearch: {
    port:            9200,  // default Elasticsearch port
    host:            "127.0.0.1", // Elasticsearch instance IP address
    path:            "/",
    indexPrefix:     "statsd",
    indexTimestamp: "day",
    countType:       "counter",
    timerType:       "timer",
    timerDataType:   "timer_data",
    gaugeDataType:   "gauge",
    formatter:       "default_format"
  }
}
```

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`

  Example:

```
STATSD_ENABLED = True
STATSD_HOST = '127.0.0.1'
STATSD_PORT = 8125
```

Map StatsD to Elasticsearch

> **Warning:** 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:

```
$ curl -X POST http://127.0.0.1:9200/_template/statsd-template -d@/opt/eclecticiq/etc-
extras/statsd/statsd_elasticsearch_template.json

# Successful response
{"acknowledged":true}
```

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 `(https://github.com/markkimsal/statsd-elasticsearch-backend)` implements a backend for Elasticsearch to work with StatsD.

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

- Go to the StatsD installation directory.
  This location may vary, depending on where exactly you installed StatsD on your system.
  For example: */opt/statsd*:

```
$ cd /opt/statsd
```

- Fetch *statsd-elasticsearch-backend*:

```
$ wget https://{user_name}:{password}@downloads.eclecticiq.com/plugin-statsd/statsd-elasticsearch-backend-0.4.2.tar.gz
```

- Invoke `npm` to install *statsd-elasticsearch-backend* as a Node.js module:

```
$ npm install statsd-elasticsearch-backend-0.4.2.tar.gz
```

Enable the service

To set up StatsD as a *systemd*-managed service, do the following:

- Go to the */etc/systemd/system* directory, and open or create the *statsd.service* file:

```
# If it already exists, open statsd.service
# Otherwise, create it:
$ nano /etc/systemd/system/statsd.service
```

- Configure the *systemd* unit file for StastD as follows:

```
[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
```

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

```
$ systemctl enable statsd
```

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

```
$ systemctl start statsd
```

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

```
$ systemctl status statsd
```

Configure Neo4j

To configure Neo4j, do the following:

- Go to */etc/neo4j*:

```
$ cd /etc/neo4j
```

Neo4j uses the following configuration files:

- */etc/neo4j/neo4j-server.properties*
- */etc/neo4j/neo4j-wrapper.conf*
- */etc/neo4j/neo4j.properties*

Check connectivity

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

Check neo4j-server.properties

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

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

- Edit the default property values, so that they reflect your system configuration.
  Example:

```
# Location of the database directory.
org.neo4j.server.database.location=/media/neo4j/graph.db

# Low-level graph engine tuning file.
org.neo4j.server.db.tuning.properties=/etc/neo4j/neo4j.properties

# The web server should listen only to the specified IP.
# Default value: 'localhost'
# (accept only local connections).
# See the security section in the neo4j manual before modifying this value.
#org.neo4j.server.webserver.address=localhost

# HTTP port (for all data, administrative, and UI access).
org.neo4j.server.webserver.port=7474

# Enable/Disable HTTPS support.
org.neo4j.server.webserver.https.enabled=false

# HTTPS port (for all data, administrative, and UI access).
org.neo4j.server.webserver.https.port=7473

# Enable/Disable auth to access Neo4j.
dbms.security.auth_enabled=false

# Certificate location (autogenerated if the file does not exist).
dbms.security.tls_certificate_file=/etc/neo4j/ssl/snakeoil.cert

# Private key location (autogenerated if the file does not exist).
dbms.security.tls_key_file=/etc/neo4j/ssl/snakeoil.key

# Internally generated keystore (don't try to put your own
# keystore there, it will get deleted when the server starts)
org.neo4j.server.webserver.https.keystore.location=data/keystore

# Comma separated list of JAX-RS packages containing JAX-RS resources, one
# package name for each mountpoint. The listed package names will be loaded
# under the mountpoints specified. Uncomment this line to mount the
# org.neo4j.examples.server.unmanaged.HelloWorldResource.java from
# neo4j-server-examples under /examples/unmanaged, resulting in a final URL of
# http://localhost:7474/examples/unmanaged/helloworld/{nodeId}
#org.neo4j.server.thirdparty_jaxrs_classes=org.neo4j.examples.server.unmanaged=/examples/unmanaged

# By default, HTTP logging is disabled.
# Set this property to 'True' to enable it.
org.neo4j.server.http.log.enabled=false

# Logging policy file controlling HTTP log output and archiving.
# Note: you can change the rollover and retention policy. However, it is
# preferable not to change the output format, since it is configured to use the
# ubiquitous common log format.
org.neo4j.server.http.log.config=/etc/neo4j/neo4j-http-logging.xml

# Servers round-robin database directory location. Allowed values:
# - absolute path such as '/var/rrd'
# - path relative to the server working directory such as 'data/rrd'
# - when commented out, it defaults to the database data directory.
org.neo4j.server.webadmin.rrdb.location=data/rrd
```

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

Inspect the amount of allocated memory: Neo4j purrs (https://www.youtube.com/watch?v=drq_ww7ytzw) when you give it plenty of memory to work with.

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

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

Check neo4j-wrapper.conf

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

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

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

```
wrapper.java.initmemory=4096
wrapper.java.maxmemory=8192
```

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

Check neo4j.properties

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

- Open the *neo4j.properties* property file:

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

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

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

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

Check permissions

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

- Create the target directory:

```
$ mkdir -p /media/neo4j
```

- Make sure the correct user owns the target directory to access it in read and write modes.
- The Neo4j installation package creates a `neo4j` user, but no group with the same name.
  To assign file or directory ownership to this user, specify `nogroup` as the associated group name: `neo4j:nogroup`.

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

Check the URL

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

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

Enable HTTPS and authentication

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.

These tasks require editing the following configuration files:

- */etc/neo4j/neo4j-server.properties*: for detailed instructions on the properties and values you need to edit to enable SSL, refer to the **Neo4j official documentation** `(https://neo4j.com/docs/2.3.8/security-server.html#security-server-https)`.

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

  The relevant properties that control secure access are:

```
# All values in this example are dummy

# Certificate location (auto-generated if the file does not exist)
dbms.security.tls_certificate_file=/etc/neo4j/ssl/snakeoil.cert

# Private key location (auto-generated if the file does not exist)
dbms.security.tls_key_file=/etc/neo4j/ssl/snakeoil.key

# Enable/Disable HTTPS
org.neo4j.server.webserver.https.enabled=true

# HTTPS port (for all data, administrative, and UI access)
org.neo4j.server.webserver.https.port=7473
```

- */opt/eclecticiq/etc/eclecticiq/platform_settings.py*: add or edit the following parameters, so that the platform is aware that communication with Neo4j flows through HTTP and SSL:

```
# 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 set in `org.neo4j.server.webserver.https.port` should be the same as the port number you assign to `NEO4J_URL`.

The SSL certificate name and path in `dbms.security.tls_certificate_file` should be the same as the value you assign to `NEO4J_SSL_VERIFY`.

Enable authentication

To enable user 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 following configuration files:

- */etc/neo4j/neo4j-server.properties*: for detailed descriptions of Neo4j authentication and authorization, refer to the Neo4j official documentation:
  - **Server authentication and authorization** `(https://neo4j.com/docs/2.3.8/security-server.html#security-server-auth)`
  - **REST API Authentication and Authorization** `(https://neo4j.com/docs/2.3.8/rest-api-security.html#rest-api-security-authenticating)`

  In *neo4j-server.properties* set the following property to `true`:

```
# Enable authorization
dbms.security.auth_enabled=true
```

- */opt/eclecticiq/etc/eclecticiq/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 specified credentials:

```
# User name to log in to Neo4j
NEO4J_USER = "neo4j"

# Password to log in to Neo4j
NEO4J_PASSWORD = "changeme"
```

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 = ""`.

Enable the service

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

```
$ systemctl enable neo4j-service
```

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

```
$ systemctl start neo4j-service
```

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

```
$ systemctl status neo4j-service
```

Configure Postfix

Postfix (http://www.postfix.org/) is the default **email server** (https://en.wikipedia.org/wiki/message_transfer_agent) shipped with the platform. If you want to enable email-based platform features like email notifications, you need to configure Postfix to handle email traffic.

Postfix configuration and setup may vary, depending on the target environment the platform is installed on. The following links point to relevant sections of the official Postfix documentation:

- **Postfix documentation oveview page** (http://www.postfix.org/documentation.html)
- **Postfix basic configuration** (http://www.postfix.org/basic_configuration_readme.html)
- **Postfix *main.cf* configuration file parameters** (http://www.postfix.org/postconf.5.html)

The platform ships with an example Postfix configuration file you can use as a customizable template: */etc/postfix/main.cf*

- The default installation location of the Postfix configuration file is */etc/postfix*.
- The Postfix configuration file is *main.cf*.

To configure Postfix as the default email server for the platform do the following:

- Open *main.cf* in a text editor:

```
$ nano /etc/postfix/main.cf
```

Example *main.cf* file:

```
myhostname = box11.platform.host
mydomain = platform.host
myorigin = $mydomain
mydestination =

relayhost = [smtp.email.server.com]:587
inet_interfaces = loopback-only

smtp_sasl_security_options = noanonymous
smtp_fallback_relay = [relay.email.server.com]
smtp_sasl_auth_enable = yes
smtp_use_tls = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_tls_policy_maps = hash:/etc/postfix/tls_policy
```

- `myhostname`: specify the **host name** (`http://www.postfix.org/postconf.5.html#myhostname`) of the platform host server where also Postix is installed.
  Example: `box11.platform.host`
- `mydomain`: specify the **internet name** (`http://www.postfix.org/postconf.5.html#mydomain`) of the platform host server where also Postix is installed.
  Example: `platform.host`
- `myorigin`: specify the **domain name** (`http://www.postfix.org/postconf.5.html#myorigin`) to use as sent-from and send-to address for local email traffic.
  Automatic email messages generated by system processes such as cronjobs use the `myorigin` value for this purpose.
  Example: `$mydomain` (domain with multiple machines), `$myhostname` (domain with one machine)
- `mydestination`: specify the **list of domains** (`http://www.postfix.org/postconf.5.html#mydestination`) the email server accepts email from.
  Example: `localhost`, `localhost.$mydomain`, `mail.$mydomain`, `www.$mydomain`, `ftp.$mydomain`
- `relayhost`: specify the **next-hop destination** (`http://www.postfix.org/postconf.5.html#relayhost`) of non-local email. It identifies the next email handler in the chain of email servers handling non-local email traffic.
  Format: [smtp.domain.name]:port
  Example: `[smtp.email.server.com]:587`
- `inet_interfaces`: specify the **network interface addresses** (`http://www.postfix.org/postconf.5.html#inet_interfaces`) Posftix receives mail on.
  Example: `all`, `loopback-only`, `127.0.01`
- `smtp_sasl_security_options`: optionally specify one or more **SASL security options** (`http://www.postfix.org/postconf.5.html#smtp_sasl_security_options`) for the SMTP client.
  Example: `noplaintext`, `noactive`, `nodictionary`, `noanonymous`, `mutual_auth`
- `smtp_fallback_relay`: optionally specify one or more **alternative relay hosts** (`http://www.postfix.org/postconf.5.html#smtp_fallback_relay`) for SMTP destinations.
  Example: `relay.email.server.com`
- `smtp_sasl_auth_enable`: enable **SASL authentication** (`http://www.postfix.org/postconf.5.html#smtp_sasl_auth_enable`).
  Example: `yes`, `no`
- `smtp_use_tls`: enable using **TLS** (`http://www.postfix.org/postconf.5.html#smtp_use_tls`) when available on the remote server.
  Example: `yes`, `no`
  Refer to the **Postfix TLS Support** (`http://www.postfix.org/tls_readme.html`) official documentation for more information.

- `smtp_sasl_password_maps`: optionally specify a lookup table with one or more **username:password entries** `(http://www.postfix.org/postconf.5.html#smtp_sasl_password_maps)` per row to identify sender, remote host name, or next-hop domain.
  Example: `username:password`
- `smtp_tls_policy_maps`: optionally specify a lookup table with one or more **SMTP client TLS security policies** `(http://www.postfix.org/postconf.5.html#smtp_sasl_password_maps)` by next-hop destination.
  Example: `[smtp.email.server.com]:587 encrypt`

Enable the service

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

```
$ systemctl enable postfix
```

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

```
$ systemctl start postfix
```

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

```
$ systemctl status postfix
```

Update the platform settings

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

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

```
"""
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
APPLICATION_PREFIX = "/api"

# Logging
LOG_LEVEL = "warning,eiq:info"
AUDIT_TRAIL_ENABLED = True
```

```
# 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 = []

# 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://<user_name>:<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-service",
    "postgresql@9.5-main"
    "redis-server",
    "statsd",
    "kibana",
]

DISCOVERY_WORKSPACE_SIZE_LIMIT = 500
```

```
DISCOVERY_SEARCH_LIMIT = 1000
DISCOVERY_RESULTS_LIMIT = 500

INGESTION_QUEUE_NAME = "queue:ingestion:inbound"
ENRICHMENT_QUEUE_NAME = "queue:enrichment:inbound"
GRAPH_QUEUE_NAME = "queue:graph:inbound"
GRAPH_INGESTION_SPOOL_DIRECTORY = "/opt/eclecticiq/tmp/gi-storage"
SEARCH_QUEUE_NAME = "queue:search:inbound"

PROXY_URL_FILE_PATH = "/opt/eclecticiq/etc/eclecticiq/proxy_url"
PLATFORM_MANIFESTS_DIR = "/opt/eclecticiq/manifests"

# 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
# The TLS and SSL values below should reflect the
# TLS and SSL settings in etc/postfix/main.cf
IS_SMTP_TLS = False
IS_SMTP_SSL = False

# 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,  # 2min
    "eiq.incoming-transports": 8 * 60 * 60,  # 8hours
    "eiq.outgoing-transports": 2 * 60,  # 2min
}

# 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": 60,
    "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. The corresponding session is terminated, and you need to sign back in to the platform.

When human interaction is detected — for example, keystrokes or mouse activity — the token is automatically refreshed every 2 minutes. This prevents the system from signing out users who may be working or saving data at that time.

Therefore, the default maximum amount of idle time without any human interaction before being automatically signed out equals to *session token validity - 2 minutes*.

- When you set the spool directory for the graph ingestion in the platform configuration file, make sure the `eclecticiq` user can access it in write mode:

```
GRAPH_INGESTION_SPOOL_DIRECTORY = '/opt/eclecticiq/tmp/gi-storage' # 'eclecticiq' user needs write rights to this dir
```

- Review the */opt/eclecticiq/etc/eclecticiq/opentaxii.yml* OpenTAXII settings to make sure they reflect your environment.

Configure OpenTAXII

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

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

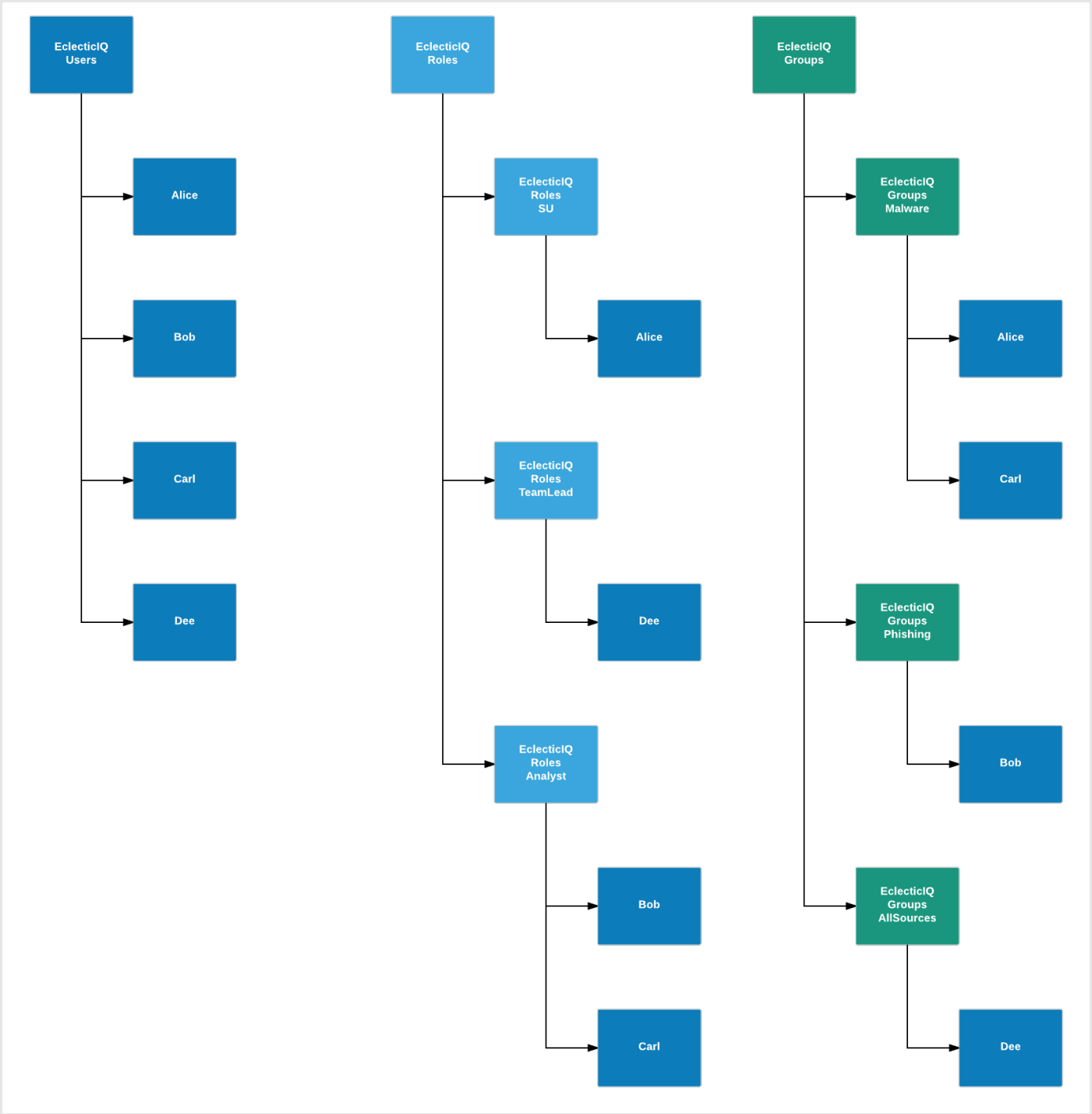
Configure LDAP authentication

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

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 = "ldaps://10.0.2.212"
LDAP_IGNORE_TLS_ERRORS = True

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

LDAP_USERS_FILTER = (
    "ou=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`
`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 - {user_name}
```

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:

```
$ 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 configuration:

```
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_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 - {user_name}
```

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.

Last generated on Oct 20, 2017

Bootstrap the platform

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

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

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 - {user_name}
```

Or:

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

```
$ sudo {command}
```

Load the Supervisor configuration

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

```
$ systemctl start supervisor
```

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

```
$ systemctl status supervisor
```

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

```
$ supervisorctl status
```

By default, Supervisor configuration files are stored in these locations:

- **Supervisor configuration file** (`http://supervisord.org/configuration.html#configuration-file`): */etc/supervisor/supervisord.conf*

- Configuration files of all Supervisor-managed applications: */etc/supervisor/*
  Make sure this location corresponds to the path defined in the `[include]` section of the *supervisord.conf* file:

```
[include]
# Default path for Ubuntu OS
files = etc/supervisor/*.ini
```

Typically, you can reach Supervisor by making a call to `http://localhost:9001` or to `http://127.0.0.1:9001`.

cURL call example:

```
$ curl http://localhost:9001
```

To make sure the platform and Supervisor can communicate, you may need to check and, if necessary, edit the *supervisord.conf* file.

Verify that the *supervisord.conf* file includes the following sections enabling the `supervisord` daemon to respond to requests from the platform:

- `[inet_http_server]` **section** `(http://supervisord.org/configuration.html#inet-http-server-section-settings)` to define the Supervisor host.
  To enable `inet_http_server`, *supervisord.conf* should contain the following properties under `[inet_http_server]`:

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

- `[rpcinterface:supervisor]` **section** `(http://supervisord.org/configuration.html#rpcinterface-x-section-settings)` to allow communication between the Supervisor XML-RPC interface and the platform API to retrieve status information.
  To enable `rpcinterface`, *supervisord.conf* should contain the following properties under `[rpcinterface:supervisor]`:

```
[rpcinterface:supervisor]
supervisor.rpcinterface_factory = supervisor.rpcinterface:make_main_rpcinterface
```

Example Supervisor configuration file:

```
[supervisord]
logfile=/var/log/supervisor/supervisord.log
logfile_maxbytes=5MB
logfile_backups=0
loglevel=info
pidfile=/var/run/supervisord.pid
nodaemon=false
minprocs=200

[unix_http_server]
file=/var/run/supervisor/supervisor.sock

[inet_http_server]
port=127.0.0.1:9001

[rpcinterface:supervisor]
supervisor.rpcinterface_factory = supervisor.rpcinterface:make_main_rpcinterface

[supervisorctl]
serverurl=unix:///var/run/supervisor/supervisor.sock

[include]
# Default path for Ubuntu OS
files = etc/supervisor/*.ini
```

To make sure the platform and Supervisor can communicate, you may need to check and, if necessary, edit the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* file.

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

```
# Supervisor service address
# Comma separated list of 'host:port'
SUPERVISORD_HOSTS = '127.0.0.1:9001'

# List of services Systemd monitors
SYSTEMD_SERVICES = [
    'elasticsearch',
    'kibana',
    'logstash',
    'neo4j-service',
    'postfix',
    'postgresql@9.5-main'
    'redis-server'
    'statsd',
]
```

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.

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

Set up the database

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

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

```
$ export EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
$ /opt/eclecticiq/migrations/create-db.sh
```

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

```
$ export 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.

- 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.

```
$ su -s /bin/bash elasticsearch -c "export
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.

Bootstrap Neo4j

To bootstrap Neo4j, do the following::

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

Prerequisites

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

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

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

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

Neo4j should be up and running:

- Check if Neo4j is running by making a cURL call to the URL defined in `NEO4J_URL` in the in the *platform_settings.py* configuration file.
  By default, it is `http://localhost:7474` on HTTP, and `http://localhost:7473` on HTTPS.
- If Neo4j is not running, restart the service by running the following command(s):

```
$ systemctl start neo4j-service
```

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

```
$ systemctl status neo4j-service
```

Create the graph schema

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

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

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

```
$ export EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
$ /opt/eclecticiq/platform/api/bin/eiq-platform graph upgrade
```

Restart Nginx

- Stop the Nginx web server:

```
$ systemctl stop nginx
```

- Start the Nginx web server:

```
$ systemctl start nginx
```

Load the dashboard

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

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

Run a final check

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

- Core processes and services
- Search, indexing and graph
- Availability
- 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 *supervisor* daemon is running, run the following command(s):

```
$ systemctl status supervisor
```

PostgreSQL

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

```
$ systemctl status postgresql@9.5-main
```

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-service
```

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
```

Reload Supervisor configurations

- After completing the installation, manually restart Supervisor by first stopping it, and then by starting it.
  In this way, it can pick up any changed configurations before starting all managed applications:

```
$ systemctl restart supervisor
```

or:

```
$ systemctl stop supervisor

$ systemctl start supervisor
```

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
```

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.

Access the platform

To access the platform, do the following:

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

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

Install platform extensions

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

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/Anubis** (https://downloads.eclecticiq.com/extensions/anubis)	incoming feed transport and content types
ArcSight	**https://downloads.eclecticiq.com/Extensions/Arcsight** (https://downloads.eclecticiq.com/extensions/arcsight)	outgoing feed content type
BFK	**https://downloads.eclecticiq.com/Extensions/Bfk** (https://downloads.eclecticiq.com/extensions/bfk)	incoming feed transport and content types
CAPEC	**https://downloads.eclecticiq.com/Extensions/Capec** (https://downloads.eclecticiq.com/extensions/capec)	incoming feed content type
Censys	**https://downloads.eclecticiq.com/Extensions/Censys** (https://downloads.eclecticiq.com/extensions/censys)	enricher
CIRCL	**https://downloads.eclecticiq.com/Extensions/Circl** (https://downloads.eclecticiq.com/extensions/circl)	enricher
Core	**https://downloads.eclecticiq.com/Extensions/Core** (https://downloads.eclecticiq.com/extensions/core)	incoming and outgoing feed transport and content types
Crowdstrike	**https://downloads.eclecticiq.com/Extensions/Crowdstrike** (https://downloads.eclecticiq.com/extensions/crowdstrike)	incoming feed, enricher
CVE Search	**https://downloads.eclecticiq.com/Extensions/CVE_search** (https://downloads.eclecticiq.com/extensions/cve_search)	incoming feed transport and content types, enricher
DomainTools	**https://downloads.eclecticiq.com/Extensions/Domaintools** (https://downloads.eclecticiq.com/extensions/domaintools)	enricher
Farsight	**https://downloads.eclecticiq.com/Extensions/Farsight** (https://downloads.eclecticiq.com/extensions/farsight)	enricher
FireEye	**https://downloads.eclecticiq.com/Extensions/Fireeye** (https://downloads.eclecticiq.com/extensions/fireeye)	incoming feed transport type, enricher

Extension	Download	Type
Flashpoint	**https://downloads.eclecticiq.com/Extensions/Flashpoint** (https://downloads.eclecticiq.com/extensions/flashpoint)	enricher
Fox-IT	**https://downloads.eclecticiq.com/Extensions/Foxit** (https://downloads.eclecticiq.com/extensions/foxit)	enricher
Intel 471	**https://downloads.eclecticiq.com/Extensions/Intel471** (https://downloads.eclecticiq.com/extensions/intel471)	incoming feed, enricher
OpenPhish	**https://downloads.eclecticiq.com/Extensions/Openphish** (https://downloads.eclecticiq.com/extensions/openphish)	incoming feed transport and content types
OpenSource	**https://downloads.eclecticiq.com/Extensions/Opensource** (https://downloads.eclecticiq.com/extensions/opensource)	-
PassiveTotal	**https://downloads.eclecticiq.com/Extensions/Passivetotal** (https://downloads.eclecticiq.com/extensions/passivetotal)	enricher
PhishMe	**https://downloads.eclecticiq.com/Extensions/Phishme** (https://downloads.eclecticiq.com/extensions/phishme)	incoming feed transport and content types
Recorded Future	**https://downloads.eclecticiq.com/Extensions/Recorded-future** (https://downloads.eclecticiq.com/extensions/recorded-future)	enricher
Reports	**https://downloads.eclecticiq.com/Extensions/Reports** (https://downloads.eclecticiq.com/extensions/reports)	outgoing feed content type
S3	**https://downloads.eclecticiq.com/Extensions/S3** (https://downloads.eclecticiq.com/extensions/s3)	incoming and outgoing feed transport type
Shodan	**https://downloads.eclecticiq.com/Extensions/Shodan** (https://downloads.eclecticiq.com/extensions/shodan)	enricher
Sighting	**https://downloads.eclecticiq.com/Extensions/Sighting** (https://downloads.eclecticiq.com/extensions/sighting)	incoming feed transport type
SpyCloud	**https://downloads.eclecticiq.com/Extensions/Spycloud** (https://downloads.eclecticiq.com/extensions/spycloud)	incoming feed transport and content types, enricher
STIX TAXII	**https://downloads.eclecticiq.com/Extensions/Stx_Taxii** (https://downloads.eclecticiq.com/extensions/stx_taxii)	incoming and outgoing feed transport and content types
Threat Grid	**https://downloads.eclecticiq.com/Extensions/Threatgrid** (https://downloads.eclecticiq.com/extensions/threatgrid)	ncoming feed transport and content types
Threat Recon	**https://downloads.eclecticiq.com/Extensions/Threatrecon** (https://downloads.eclecticiq.com/extensions/threatrecon)	incoming feed content type

Extension	Download	Type
VirusTotal	**https://downloads.eclecticiq.com/Extensions/Virustotal** (https://downloads.eclecticiq.com/extensions/virustotal)	enricher
Wapack Labs	**https://downloads.eclecticiq.com/Extensions/Wapacklabs** (https://downloads.eclecticiq.com/extensions/wapacklabs)	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}-{extension_version}.tar.gz*
  Example: *eclecticiq-extension-anubis-1.0.tar.gz*
- 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}-{extension_version}.tar.gz

# Example:
$ pip install eclecticiq-extension-anubis-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 supervisor

$ 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.

Upgrade the platform

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

When you download a new *.deb* 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 - {user_name}
```

Or:

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

```
$ sudo {command}
```

The upgrade procedure consists of the following steps:

Before the upgrade

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

During the upgrade

- Install the new package

After the upgrade

- Check the configuration
- Check third-party configurations
- Migrate the database
- Check component versions
- Run the fixtures

- Run a final check
- Reload Supervisor configurations
- Install extensions
- Access the platform

Exit the platform

Sign out of the platform:

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

Back up your data

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

Shut down the platform

To 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@9.5-main redis-server neo4j-service 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. Usually, PID files are stored under *var/run*.
- Stop systemd-managed services:
  - *postfix*
  - *redis-server*
  - *statsd*
  - *kibana*
  - *neo4j-service*
  - *elasticsearch*
  - *postgresql@9.5-main*

```
$ systemctl stop postfix redis-server statsd kibana neo4j-service elasticsearch postgresql@9.5-
main
```

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 the following conditions are satisfied:

- All required third-party components are already installed on the target system, and their versions match the recommended version information provided.

Remove deprecated packages

Before installing the new platform release, you may need to remove any deprecated packages.
When applicable, deprecated package notifications are included in this section, as well as in the product release notes for the release they refer to.
To remove deprecated packages, run the following command(s):

```
$ apt-get remove <package_name>

$ apt-get purge <package_name>
```

Install the new package

Before beginning the upgrade procedure, make sure you have the necessary login credentials to access the provided resource download locations on Bintray.

Add key and repository

- Download the PGP public key and add it to the list by running the following command(s):

```
$ wget -qO- https://downloads.eclecticiq.com/public/GPG-KEY-eclecticiq | apt-key add -

# Successful response
OK
```

- Add the EclecticIQ Platform repository to the list of configured sources by running the following command(s):

```
$ echo 'deb https://{user_name}:{password}@downloads.eclecticiq.com/eiq-platform-deb xenial
2.0.1' > /etc/apt/sources.list.d/eclecticiq-platform.list
```

- Download and update package information for all configured sources:

```
$ apt-get update
```

Upgrade from the APT repository

- Upgrade from the previous EclecticIQ Platform release by running the following command(s):

```
$ apt-get --only-upgrade install eclecticiq-platform
```

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

It is possible to upgrade the platform only sequentially, it is not possible to skip versions.
For example, if you want to upgrade from release 1.10.0 to 1.14.1, you need to upgrade step by step by installing all intermediate releases until 1.14.1 included.

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

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

exit 99
```

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

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

```
You are almost there, just a few more steps before you complete the EclecticIQ Platform upgrade
procedure.

Refer to the upgrade section in the EclecticIQ Platform 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.

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/)`.

Migrate the database

Before starting with the steps described in this section, make sure that systems and processes are running, as described in detail in the configuration and more briefly in the bootstrapping sections.

Migrate PostegreSQL

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

Time to migrate the database.

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

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

```
$ su - eclecticiq
```

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

Reindex Elasticsearch

To reindex 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.
- `reindex_elasticsearch` 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.

```
$ su -s /bin/bash elasticsearch -c "export
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.

```
$ su -s /bin/bash elasticsearch -c "export
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.

Migrate the graph database

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

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

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

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

```
$ export EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
$ /opt/eclecticiq/platform/api/bin/eiq-platform graph upgrade
```

Check component versions

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

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

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

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

```
{
  "data": {

    "platform-api": {
      "branch": "master",
      "source": "github",
      "summary": "release/2.0dev-123-abc1234d",
      "tag": "latest",
      "version": "fb1234c2a62be5a409b38bedf65k273psgf99h54"
    },

    "platform-database": {
      "current": "109d2b29ead",
      "head": "109d2b29ead (head)",
      /*
        Allowed status values:
        - Up to date
        - Outdated
        - Head not available (when no manifest file is found)
      */
      "status": "Up to date"
    },

    "platform-ui": {
      "branch": "master",
      "source": "github",
      "summary": "release/2.0-tp11605-102-g5a04384",
      "tag": "latest",
      "version": "5a04384798f7fa5e9a4a888ss863f77e42abc66d"
    }
  }
}
```

The version information in the API JSON response matches the corresponding details in the manifest files. Manifest files are stored in the following directory:

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

The directory contains these manifest files:

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

After migrating the database, the current database version needs to match the one in the *platform-database.mf* manifest file.
To perform this check, follow the steps described below.

- Request the current database version:

```
$ export EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
$ cd /opt/eclecticiq/migrations/
$ /opt/eclecticiq/platform/api/bin/eiq-platform database current-version
```

Example:

```
109d2b29ead (head)
```

- Go to the manifest directory:

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

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

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

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

```
head: 109d2b29ead (head)
```

Run the fixtures

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

```
$ export EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
$ /opt/eclecticiq/migrations/load-fixtures.sh
```

Run a final check

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

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

- 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 *supervisor* daemon is running, run the following command(s):

```
$ systemctl status supervisor
```

PostgreSQL

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

```
$ systemctl status postgresql@9.5-main
```

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-service
```

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
```

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
```

Rewire observables to v.2.0

Rewiring observables to v.2.0 is a one-off task you need to perform only when upgrading EclecticIQ Platform from v.1.14.x to 2.0.

After successfully upgrading EclecticIQ Platform from version 1.14.x to 2.0, you need to run `eiq-platform observable refresh`.

Run this script only after booting up the platform, and after starting all platform components. The platform needs to be fully up and running for the observable refresh script to work correctly.

From v.2.0 observables are powerful tools to drive IOC-centric analysis. Observables ingested and created with previous versions of the platform need to be rewired and upgraded to v.2.0, so that the platform can, among others, index them and make them searchable.

The script reparses existing observables to update their format to platform 2.0-compliant.

Run the script from the terminal or the command line:

```
$ export EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
$ /opt/eclecticiq/platform/api/bin/eiq-platform observable refresh --workers=4
```

`workers`

It is a mandatory parameter.
It takes an integer as a value.
If you do not specify any value, it defaults to `1`.

It defines the number of workers the script should concurrently run in parallel.
The process is resource-intensive, and it takes some time to complete: gauge the number of workers based on your system resources.
`4` workers is a rule-of-thumb value to run the script with a limited impact on normal platform operation.

Logging

- After initializing, the script starts discovering items to process, and outputting log information to the terminal.
- After successfully completing, it notifies you with the following message: `all entities processed`.

Errors

- If the user terminates the script before it completes, the script returns `Aborted!`.
- If an error occurs during execution, the script returns `an unexpected worker error occurred`, along with traceback information.

In case of errors, you may want to check the traceback information, as well as the `first_entity_id` and `last_entity_id` values in the log block preceding the error message: they correspond to the first and last entities in the last successfully processed batch before the error.

Example:

```
{"event": "sending batch to search", "first_entity_id": "00209576-bd04-48be-a4f0-c9a865b2b0c0",
"last_entity_id": "0024eceb-fb14-44ee-8e8d-6dad0ce58849", "level": "info", "logger":
"eiq.platform.scripts.bulk_refresh_extracts", "timestamp": "2017-09-01T16:26:44.685313Z",
"worker_pid": 9332}
```

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.

Access the platform

To access the platform, do the following:

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

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

Backup guidelines

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

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

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

Before starting a platform backup, verify the following points:

- No users are signed in to the platform;
- No read/write activity is in progress 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 - {user_name}
```

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

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

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

# Logstash heap size
/opt/eclecticiq/etc/sysconfig/logstash # CentOS, RHEL
/opt/eclecticiq/etc/default/logstash # Ubuntu

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

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

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

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

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

# Nginx web server config file
/etc/nginx/nginx.conf
/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
```

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. Usually, PID files are stored under *var/run*.
- Stop systemd-managed services:
  - *postfix*
  - *redis-server*
  - *statsd*
  - *kibana*
  - *neo4j-service*
  - *elasticsearch*
  - *postgresql@9.5-main*

```
$ systemctl stop postfix redis-server statsd kibana neo4j-service elasticsearch postgresql@9.5-main
```

Back up the PostgreSQL database

You can back up a PostgreSQL database in several ways:

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

SQL dump

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

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

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

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

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

File system level backup

File system level backup (http://www.postgresql.org/docs/current/static/backup-file.html) creates backup copies of the PostgreSQL file used to store the database data corpus. The files to back up are stored in the **database cluster** (http://www.postgresql.org/docs/current/static/creating-cluster.html) specified with the

initdb (http://www.postgresql.org/docs/current/static/app-initdb.html) command during database storage location inizialization.
This approach requires database downtime.

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

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

```
$ cd /etc/postgresql/9.5/main/
```

- 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 /etc/postgresql/9.5/main/*
```

- 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 /etc/postgresql/9.5/main/
```

- 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-service
```

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

```
$ systemctl status neo4j-service
```

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

```
org.neo4j.server.database.location=/media/neo4j/graph.db
```

Data recovery

To restore backed up data, follow the standard recommendations and procedures for PostgreSQL, Elasticsearch, and Neo4j:

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

Reindex Elasticsearch

To reindex 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.
- `reindex_elasticsearch` 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.

```
$ su -s /bin/bash elasticsearch -c "export
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.

```
$ su -s /bin/bash elasticsearch -c "export
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.