Lab 8. Elastic X-Pack

X-Pack is an Elastic Stack extension that bundles security, alerting, monitoring, reporting, machine learning, and graph capabilities into one easy-to-install package. It adds essential features to make Elastic Stack production-ready. Unlike the components of Elastic Stack, which are open source, X-Pack is a commercial offering from Elastic.co, and so it requires a paid license so that it can be used. When you install X-Pack for the first time, you are given a 30-day trial. Even though X-Pack comes as a bundle, it allows you to easily enable or disable the features you want to use.In this lab, we won't be covering all the features provided by X-Pack, but we will be covering the important features that an X-Pack beginner should be aware of.

In this lab, we will cover the following topics:

- Installing Elasticsearch and Kibana with X-Pack
- Configuring/activating X-Pack trial account
- Securing Elasticsearch and Kibana
- Monitoring Elasticsearch
- Alerting

Installing Elasticsearch and Kibana with X-Pack

Prior to Elastic 6.3, X-Pack was an extension of Elastic Stack that could have been installed on top of Elasticsearch or Kibana. Now, Elasticsearch and Kibana come in two flavors:

- OSS version, that is, Apache 2.0License
- Elastic license

If X-Pack isn't available with the OSS version, then you have to download Elasticsearch and Kibana with the Elastic license. When you download Elasticsearch or Kibana with the Elastic license, all the basic features are available for free by default. The paid features can be used as a trial for 30 days; post that time period, a license has to be bought.

Note

To see a list of all the features that are available for free and the features that are paid, go to https://www.elastic.co/subscriptions.

Installation

In order to explore X-Pack and its features, we will need to download Elasticsearch and Kibana with Elastic license.

You can download Elasticsearch from https://www.elastic.co/downloads/past-releases/elasticsearch-7-0-0.

You can download Kibana from https://www.elastic.co/downloads/past-releases/kibana-7-0-0.

Note

Before installation, please make sure to stop any existing running instances of Elastic Stack components.

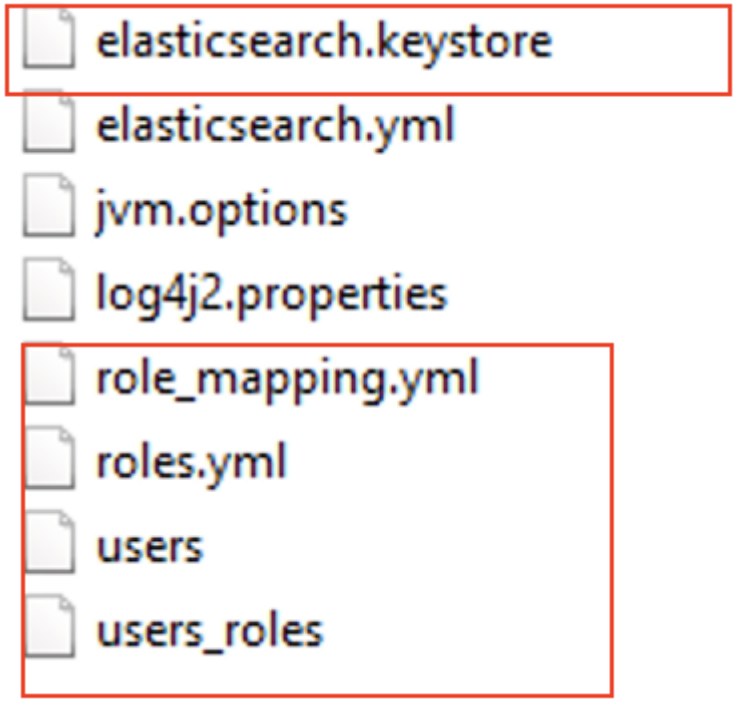
When you start Elasticsearch, you should see X-Pack-related plugins getting loaded and new files getting created under the `ES_HOME/config` folder. The bootup logs should also indicate the license type; in this case, this will be `basic`:

```
[2019-05-21T13:38:08,679][INFO ][o.e.p.PluginsService ] [MADSH01-APM01] loaded module
[x-pack-ccr]
[2019-05-21T13:38:08,682][INFO ][o.e.p.PluginsService ] [MADSH01-APM01] loaded module
[x-pack-core]
[2019-05-21T13:38:08,684][INFO ][o.e.p.PluginsService ] [MADSH01-APM01] loaded module
```

```
[x-pack-deprecation]
[2019-05-21T13:38:08,687][INFO ][o.e.p.PluginsService ] [MADSH01-APM01] loaded module
[x-pack-graph]
[2019-05-21T13:38:08,689][INFO ][o.e.p.PluginsService ] [MADSH01-APM01] loaded module
[x-pack-ilm]
[2019-05-21T13:38:08,690][INFO ][o.e.p.PluginsService ] [MADSH01-APM01] loaded module
[x-pack-logstash]
[2019-05-21T13:38:08,692][INFO ][o.e.p.PluginsService ] [MADSH01-APM01] loaded module
[x-pack-ml]
[2019-05-21T13:38:08,694][INFO ][o.e.p.PluginsService ] [MADSH01-APM01] loaded module
[x-pack-monitoring]
[2019-05-21T13:38:08,696][INFO ][o.e.p.PluginsService ] [MADSH01-APM01] loaded module
[x-pack-rollup]
[2019-05-21T13:38:08,698][INFO ][o.e.p.PluginsService ] [MADSH01-APM01] loaded module
[x-pack-security]
[2019-05-21T13:38:08,700][INFO ][o.e.p.PluginsService ] [MADSH01-APM01] loaded module
[x-pack-sql]
[2019-05-21T13:38:08,707][INFO ][o.e.p.PluginsService ] [MADSH01-APM01] loaded module
[x-pack-watcher]

:
:
[2019-05-21T13:38:18,783][INFO ][o.e.l.LicenseService ] [MADSH01-APM01] license
[727c7d40-2008-428f-bc9f-b7f511fc399e] mode [basic] - valid
[2019-05-21T13:38:18,843][INFO ][o.e.g.GatewayService ] [MADSH01-APM01] recovered [0]
indices into cluster_states
```

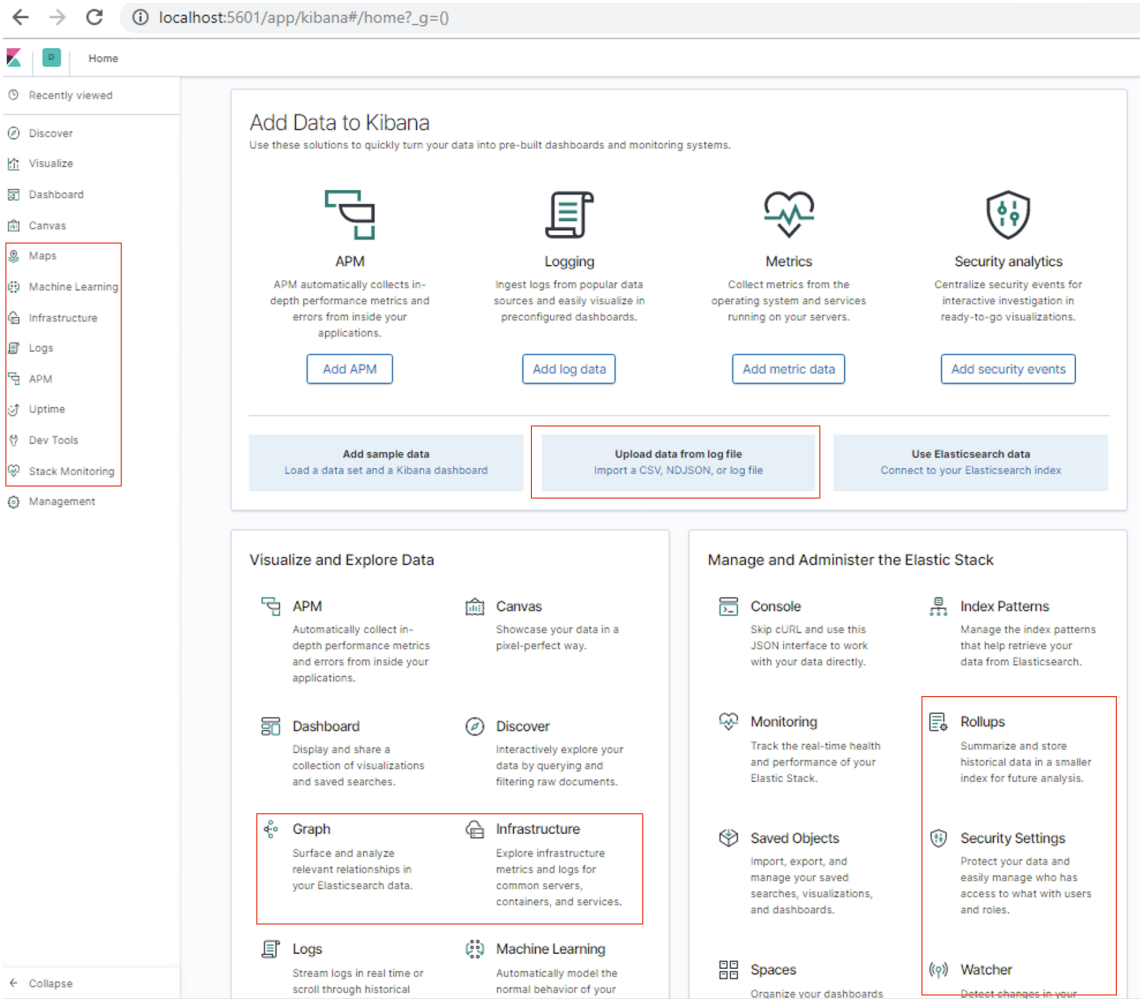
If you list the files under the `ES_HOME/config` directory, you should see the files being displayed, as shown in the following screenshot. All the highlighted files are the new files that aren't available if the Elasticsearch OSS version is used:



When you start Kibana, you should see a list of X-Pack-related plugins that are loaded, as shown in the following code:

```
  log [05:57:13.939] [info][status][plugin:xpack_main@7.0.0] Status changed from
uninitialized to yellow - Waiting for Elasticsearch
  log [05:57:13.952] [info][status][plugin:graph@7.0.0] Status changed from
uninitialized to yellow - Waiting for Elasticsearch
  log [05:57:13.968] [info][status][plugin:monitoring@7.0.0] Status changed from
uninitialized to green - Ready
  log [05:57:13.974] [info][status][plugin:spaces@7.0.0] Status changed from
uninitialized to yellow - Waiting for Elasticsearch
  :
  :
  log [05:57:14.136] [info][status][plugin:beats_management@7.0.0] Status changed from
uninitialized to yellow - Waiting for Elasticsearch
  log [05:57:14.162] [info][status][plugin:apm_oss@undefined] Status changed from
uninitialized to green - Ready
  log [05:57:14.179] [info][status][plugin:apm@7.0.0] Status changed from
uninitialized to green - Ready
 :
 :
log [05:57:16.798] [info][license][xpack] Imported license information from
Elasticsearch for the [data] cluster: mode: basic | status: active
  :
  :
  log [05:57:19.711] [info][listening] Server running at http://localhost:5601
  log [05:57:20.530] [info][status][plugin:spaces@7.0.0] Status changed from yellow to
green - Ready
```

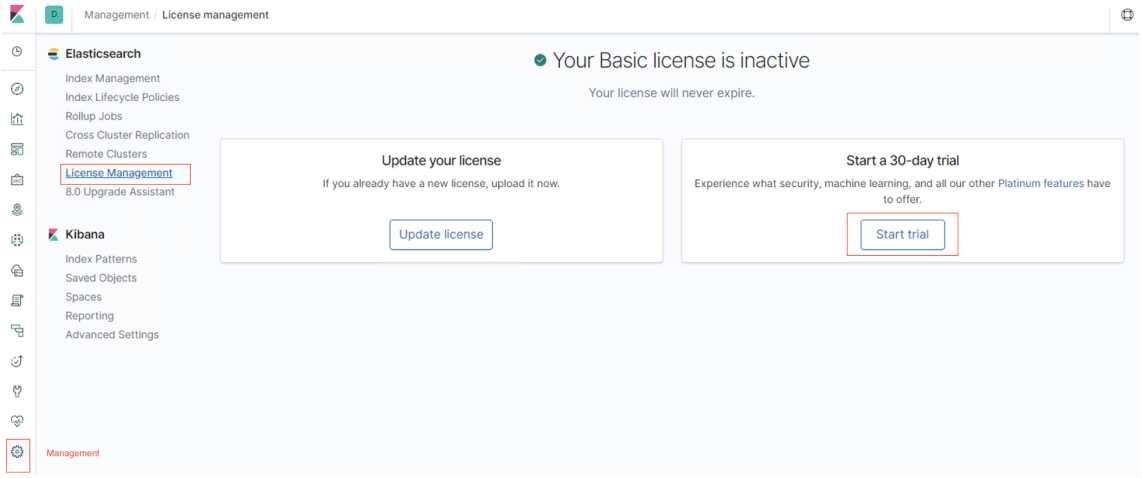
Open Kibana by navigating to `http:localhost:5601`. You should see the following screen. Some of the new features that are not present in the OSS version are highlighted in the following screenshot:



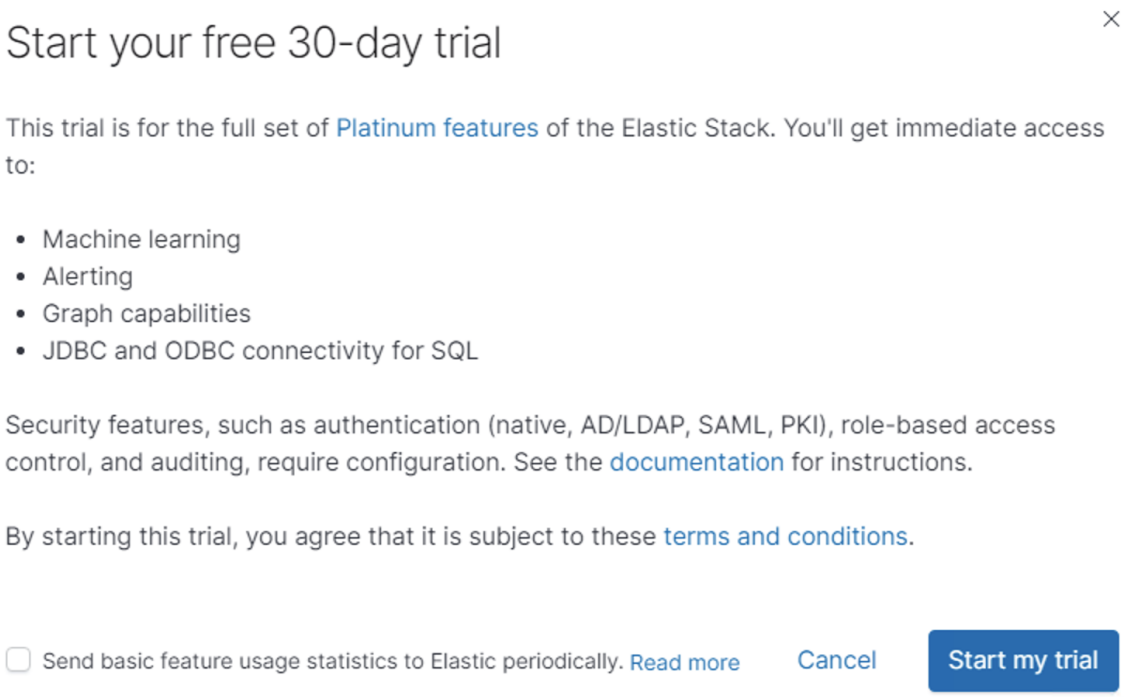
Activating X-Pack trial account

In order to activate all the X-Pack paid features, we need to enable the trial account, which is valid for 30 days. Let's go ahead and activate it.

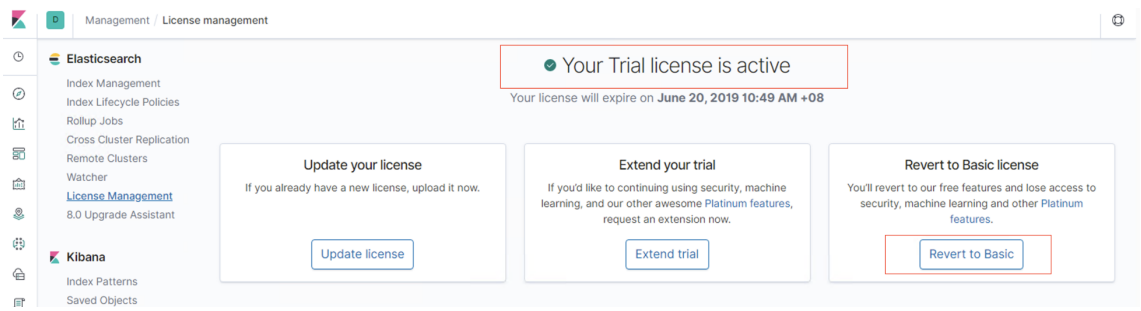
Click on the `Management` icon on the left-hand side menu and click on `License Management`. Then, click on `Start Trial`, as follows:



Click on the `Start my Trial` button in the resultant popup, as follows:



On successful activation, you should see the status of the license as `Active`. At any point in time before the trial ends, you can go ahead and revert back to the basic license by clicking on the `Revert to Basic` button:



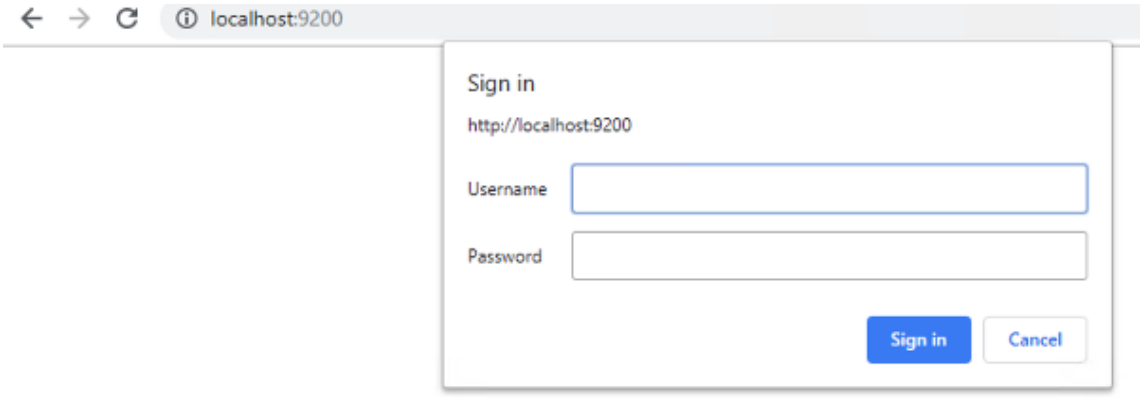
Open `elasticsearch.yml`, which can be found under the `$ES_HOME/config` folder, and add the following line at the end of the file to enable X-Pack and restart Elasticsearch and Kibana:

```
xpack.security.enabled: true
```

Note

If the Elasticsearch cluster has multiple nodes, then the `xpack.security.enabled: true` property must be set in each of the nodes and restarted.

Now, when you try to access Elasticsearch via `http://localhost:9200`, the user will be prompted for login credentials, as shown in the following screenshot:



Similarly, if you see the logs of Kibana in the Terminal, it would fail to connect to Elasticsearch due to authentication issues and won't come up until we set the right credentials in the `kibana.yml` file.

Go ahead and stop Kibana. Let Elasticsearch run. Now that X-Pack is enabled and security is in place, how do we know what credentials to use to log in? We will look at this in the next section.

Generating passwords for default users

Elastic Stack security comes with default users and a built-in credential helper to set up security with ease and have things up and running quickly. Open up another Terminal and navigate to `ES_HOME`. Let's generate the passwords for the reserved/default users---`elastic`, `kibana`, `apm_system`, `remote_monitoring_user`, `beats_system`, and `logstash_system`---by executing the following command:

```
$ ES_HOME>bin/elasticsearch-setup-passwords interactive
```

You should get the following logs/prompts to enter the password for the reserved/default users:

```
Initiating the setup of passwords for reserved users
elastic,apm_system,kibana,logstash_system,beats_system,remote_monitoring_user.
You will be prompted to enter passwords as the process progresses.
Please confirm that you would like to continue [y/N]y

Enter password for [elastic]:elastic
Reenter password for [elastic]:elastic
Enter password for [apm_system]:apm_system
Reenter password for [apm_system]:apm_system
Enter password for [kibana]:kibana
Reenter password for [kibana]:kibana
Enter password for [logstash_system]:logstash_system
Reenter password for [logstash_system]:logstash_system
Enter password for [beats_system]:beats_system
Reenter password for [beats_system]:beats_system
Enter password for [remote_monitoring_user]:remote_monitoring_user
Reenter password for [remote_monitoring_user]:remote_monitoring_user
Changed password for user [apm_system]
Changed password for user [kibana]
Changed password for user [logstash_system]
Changed password for user [beats_system]
Changed password for user [remote_monitoring_user]
Changed password for user [elastic]
```

Note

Please make a note of the passwords that have been set for the reserved/default users. You can choose any password of your liking. We have chosen the passwords as `elastic`, `kibana`, `logstash_system`, `beats_system`, `apm_system`, and `remote_monitoring_user` for `elastic`, `kibana`, `logstash_system`, `beats_system`, `apm_system`, and `remote_monitoring_user` users, respectively, and we will be using them throughout this lab.

Note

All the security-related information for the built-in users will be stored in a special index called `.security` and will be managed by Elasticsearch.

To verify X-Pack's installation and enforcement of security, point your web browser to `http://localhost:9200/` to open Elasticsearch. You should be prompted to log in to Elasticsearch. To log in, you can use the built-in `elastic` user and `elastic` password. Upon logging in, you should see the following response:

```
{
  "name" : "MADSH01-APM01",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "I2RVLSk2Rr6IRJb6zDf19g",
  "version" : {
    "number" : "7.0.0",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "b7e28a7",
    "build_date" : "2019-04-05T22:55:32.697037Z",
```

```
    "build_snapshot" : false,
    "lucene_version" : "8.0.0",
    "minimum_wire_compatibility_version" : "6.7.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Before we can go ahead and start Kibana, we need to set the Elasticsearch credentials in `kibana.yml` so that when we boot up Kibana, it knows what credentials it needs to use for authenticating itself/communicating with Elasticsearch.

Add the following credentials in the `kibana.yml` file, which can be found under `$KIBANA_HOME/config`, and save it:

```
elasticsearch.username: "kibana"
elasticsearch.password: "kibana"
```

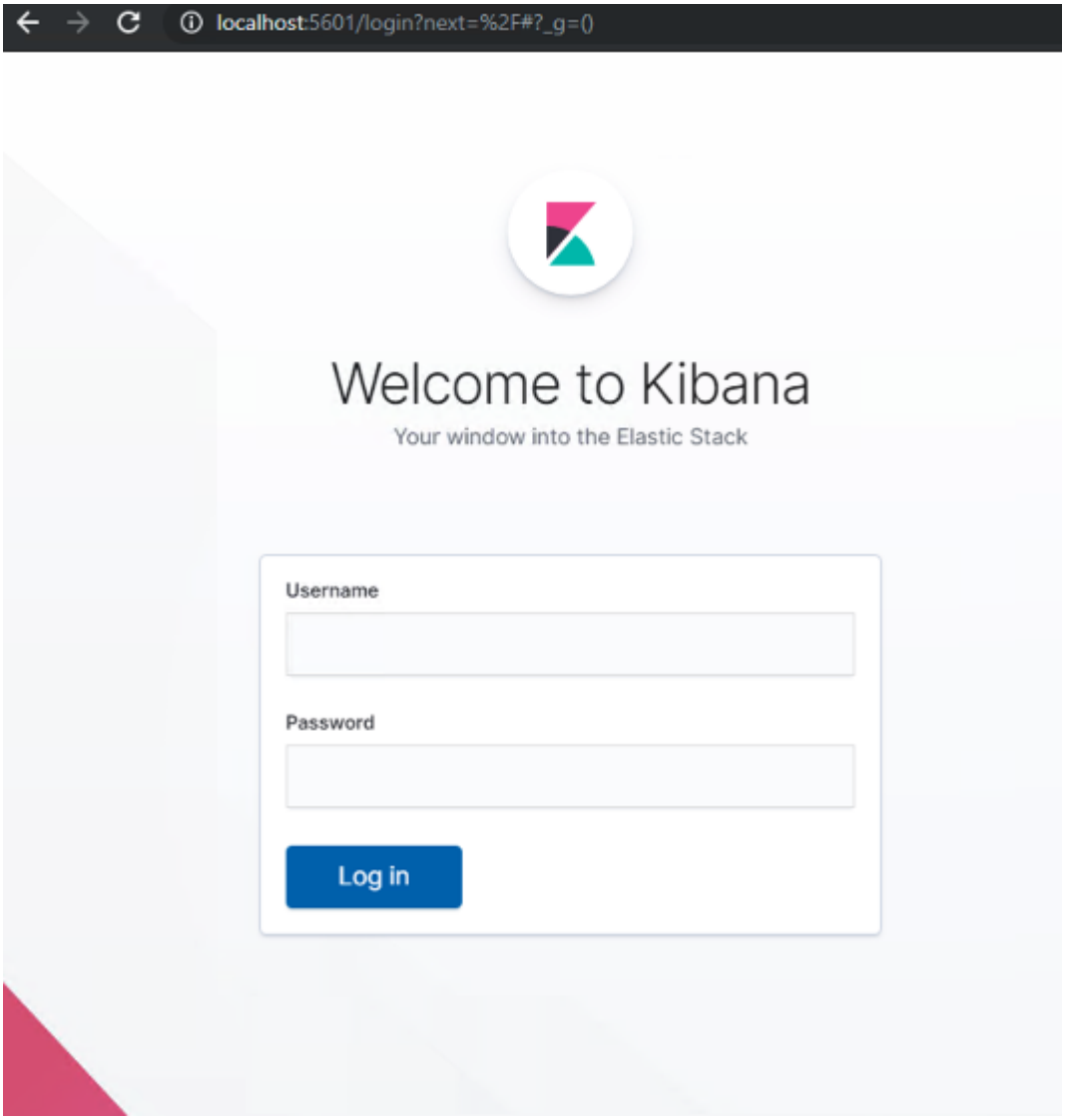
Note

If you have chosen a different password for the `kibana` user during password setup, use that value for the `elasticsearch.password` property.

Start Kibana:

```
$KIBANA_HOME>bin/kibana
```

To verify that the authentication is in place, go to `http://localhost:5601/` to open Kibana. You should be prompted to login to Kibana. To log in, you can use the built-in `elastic` user and the `elastic` password, as follows:



Note

The built-in `kibana` user will be used to connect and communicate with Elasticsearch. Each built-in user has a specific role which provides certain authorization and restrictions for specific activities. It is recommended to use the `kibana` user for this. We will be covering roles in more detail in the upcoming sections.

Configuring X-Pack

X-Pack comes bundled with security, alerting, monitoring, reporting, machine learning, and graph capabilities. By default, all of these features are enabled. However, you might not be interested in all of the features it provides. You can selectively enable and disable the features that you are interested in from the `elasticsearch.yml` and `kibana.yml` configuration files.

Elasticsearch supports the following features and settings in the `elasticsearch.yml` file:

Feature	Setting	Description
Machine learning	`xpack.ml.enabled`	Set this to false to disable X-Pack machine learning features
Monitoring	`xpack.monitoring.enabled`	Set this to false to disable Elasticsearch's monitoring features
Security	`xpack.security.enabled`	Set this to false to disable X-Pack security features
Watcher	`xpack.watcher.enabled`	Set this to false to disable Watcher

Kibana supports the following features and settings in the `kibana.yml` file:

Feature	Setting	Description
Machine learning	`xpack.ml.enabled`	Set this to false to disable X-Pack machine learning features
Monitoring	`xpack.monitoring.enabled`	Set this to false to disable Kibana's monitoring features
Security	`xpack.security.enabled`	Set this to false to disable X-Pack security features
Graph	`xpack.graph.enabled`	Set this to false to disable X-Pack graph features
Reporting	`xpack.reporting.enabled`	Set this to false to disable X-Pack reporting features

Note

If X-Pack is installed on Logstash, you can disable monitoring by setting the `xpack.monitoring.enabled` property to false in the `logstash.yml` configuration file.

Securing Elasticsearch and Kibana

The components of Elastic Stack are unsecured, as it doesn't have inherent security built into it; this means it can be accessed by anyone. This poses a security risk when running Elastic Stack in production. In order to prevent unauthorized access in production, different mechanisms of imposing security, such as running Elastic Stack behind a firewall and securing via reverse proxies (such as nginx, HAProxy, and so on), are employed. Elastic.co offers a commercial product to secure Elastic Stack. This offering is part of X-Pack and the module is called **Security**.

The X-Pack security module provides the following ways to secure Elastic Stack:

- User authentication and user authorization
- Node/Client authentication and channel encryption
- Auditing

User authentication

User authentication is the process of validating the user and thus preventing unauthorized access to the Elastic Cluster. In the X-Pack security module, the authentication process is handled by one or more authentication services

called **realms**. The `Security` module provides two types of realms, namely internal realms and external realms.

The two types of built-in internal realms are `native` and `file`. The `native` realm is the default realm, and the user credentials are stored in a special index called `.security-7` on Elasticsearch itself. These users are managed using the **User Management API** or the `Management` page of the Kibana UI. We will be exploring this in more detail later in this lab.

If the realm is of the `file` type, then the user credentials are stored in a file on each node. These users are managed via dedicated tools that are provided by X-Pack. These tools can be found at `$ES_HOME\bin\`. The files are stored under the `$ES_HOME\config` folder. Since the credentials are stored in a file, it is the responsibility of the administrator to create users with the same credentials on each node.

The built-in external realms are `ldap`, `active_directory`, and `pki`, which use the external LDAP server, the external Active Directory Server, and the Public Key Infrastructure, respectively, to authenticate users.

Depending on the realms that have been configured, the user credentials need to be attached to the requests that are sent to Elasticsearch. Realms live within a realm chain. The realm's order is configured in the `elasticsearch.yml` file and determines the order in which realms are consulted during the authentication process. Each realm is consulted one by one based on the order defined until the authentication is successful. Once one of the realms successfully authenticates the request, the authentication is considered to be successful. If none of the realms are able to authenticate the user, then the authentication is considered unsuccessful and an authentication error (HTTP 401) will be returned to the caller. The default realm chain consists of internal realm types, that is, `native` and `file`.

If none of these realms are specified in `elasticsearch.yml`, then the default realm that's used is `native`. To use the `file` type realm or external realms, they need to be specified in the `elasticsearch.yml` file.

For example, the following snippet shows the configuration for the realm chain containing `native`, `file`, and `ldap`:

```
xpack.security.authc:
  realms:
    native:
      type: native
      order: 0

    file:
      type: file
      order: 1

    ldap_server:
      type: ldap
      order: 2
      url: 'url_to_ldap_server'
```

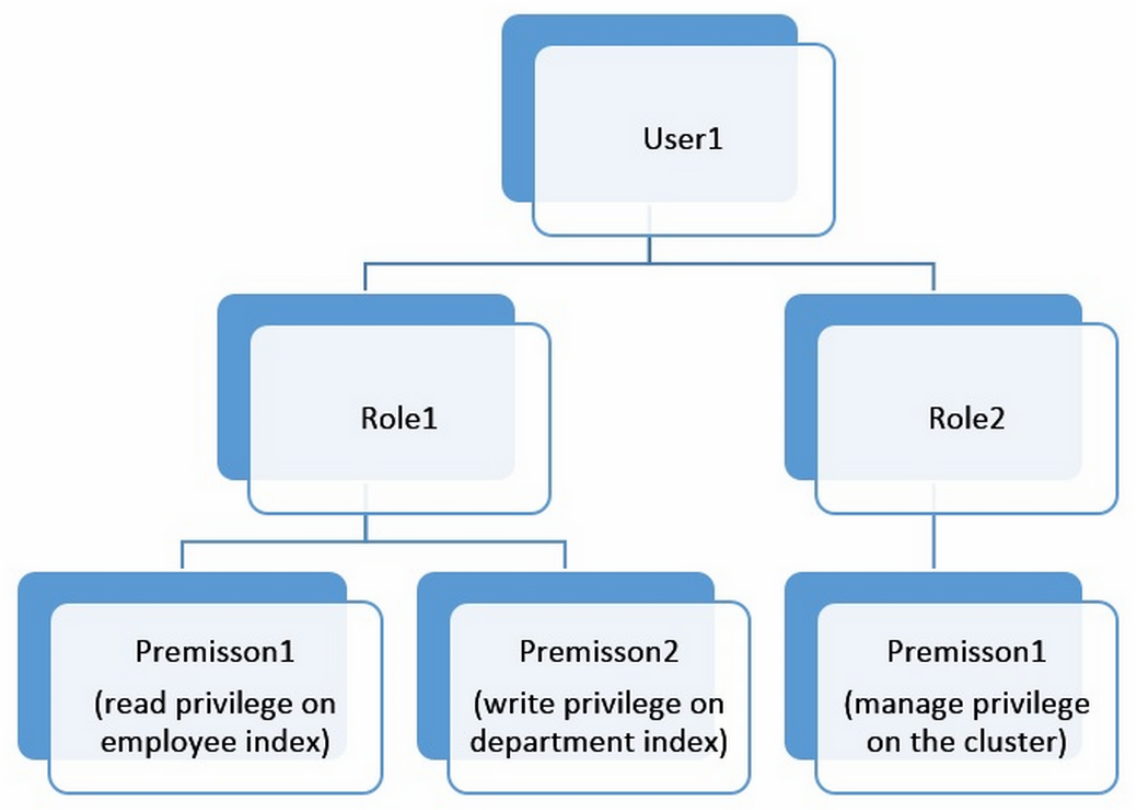
Note

To disable a specific realm `type`, use the `enabled:false` property, as shown in the following example: `ldap_server: type: ldap order: 2 enabled: false url: 'url_to_ldap_server'`

User authorization

Once the user has been successfully authenticated, the authorization process kicks in. Authorization determines whether the user behind the request has enough permissions to execute a particular request.

In X-Pack security, **secured resources** are the foundation of user-based security. A secured resource is a resource that needs access, such as indexes, documents, or fields, to perform Elasticsearch cluster operations. X-Pack security enables authorization by assigning permissions to roles that are assigned to users. A permission is one or more privileges against a secured resource. A privilege is a named group representing one or more actions that a user may execute against a secured resource. A user can have one or more roles, and the total set of permissions that a user has is defined as a union of the permissions in all its roles, as shown in the following diagram:



The X-Pack security module provides three types of privileges:

- **Cluster privileges**: Cluster privileges provide privileges for performing various operations on the cluster:
  - `all` : Allows you to execute cluster administration operations settings, as well as update, reroute, or manage users and roles
  - `monitor` : Allows you to execute all cluster read-only operations, such as fetching cluster health, cluster state, nodes' state, and more, for monitoring purposes
  - `manage` : This allows you to execute and perform cluster operations that can update the cluster, such as rerouting and updating cluster settings
- **Index privileges**: Index privileges provide privileges for performing various operations on indexes:
  - `all` : Allows you to execute any operation on an index
  - `read` : Allows you to execute read-only operations on an index, such as invoking search, get, suggest, and many more APIs
  - `create_index` : This privilege allows you to create a new index
  - `create` : This privilege allows you to index new documents into an index

- **Run As privilege**: This provides the ability to perform user impersonation; that is, it allows an authenticated user to test out another users' access rights without knowing their credentials.

Note

A complete list of all the privileges can be obtained at https://www.elastic.co/guide/en/elastic-stack-overview/7.0/security-privileges.html.

- **Node/client authentication and channel encryption**: By encrypting the communication, X-Pack security prevents network-based attacks. It provides you with the ability to encrypt traffic to and from the Elasticsearch cluster to outside applications, as well as encrypt the communication between nodes in the cluster. To prevent unintended nodes from joining the cluster, you can configure the nodes to authenticate as they join the cluster using SSL certificates. X-Pack security IP filtering can prevent unintended application clients, node clients, or transport clients from joining the cluster.
- **Auditing**: Auditing allows you to capture suspicious activity in your cluster. You can enable auditing to keep track of security-related events, such as authentication failures and refused connections. Logging these events allows you to monitor the cluster for suspicious activity and provides evidence in the event of an attack.

Security in action

In this section, we will look into creating new users, creating new roles, and associating roles with users. Let's import some sample data and use it to understand how security works.

Save the following data to a file named `data.json`:

```
{"index" : {"_index":"employee"}}
{ "name":"user1", "email":"user1@fenago.com","salary":5000, "gender":"M",
"address1":"312 Main St", "address2":"Walthill", "state":"NE"}
{"index" : {"_index":"employee"}}
{ "name":"user2", "email":"user2@fenago.com","salary":10000, "gender":"F",
"address1":"5658 N Denver Ave", "address2":"Portland", "state":"OR"}
{"index" : {"_index":"employee"}}
{ "name":"user3", "email":"user3@fenago.com","salary":7000, "gender":"F",
"address1":"300 Quinterra Ln", "address2":"Danville", "state":"CA"}
{"index" : {"_index":"department","_type":"department"}}
{ "name":"IT", "employees":50 }
{"index" : {"_index":"department","_type":"department"}}
{ "name":"SALES", "employees":500 }
{"index" : {"_index":"department","_type":"department"}}
{ "name":"SUPPORT", "employees":100 }
```

Note

The `_bulk` API requires the last line of the file to end with the newline character, `\n`. While saving the file, make sure that you have a newline as the last line of the file.

Navigate to the directory where the file is stored and execute the following command to import the data into Elasticsearch:

```
$ directoy_of_data_file> curl -s -H "Content-Type: application/json" -u
elastic:elastic -XPOST http://localhost:9200/_bulk --data-binary @data.json
```

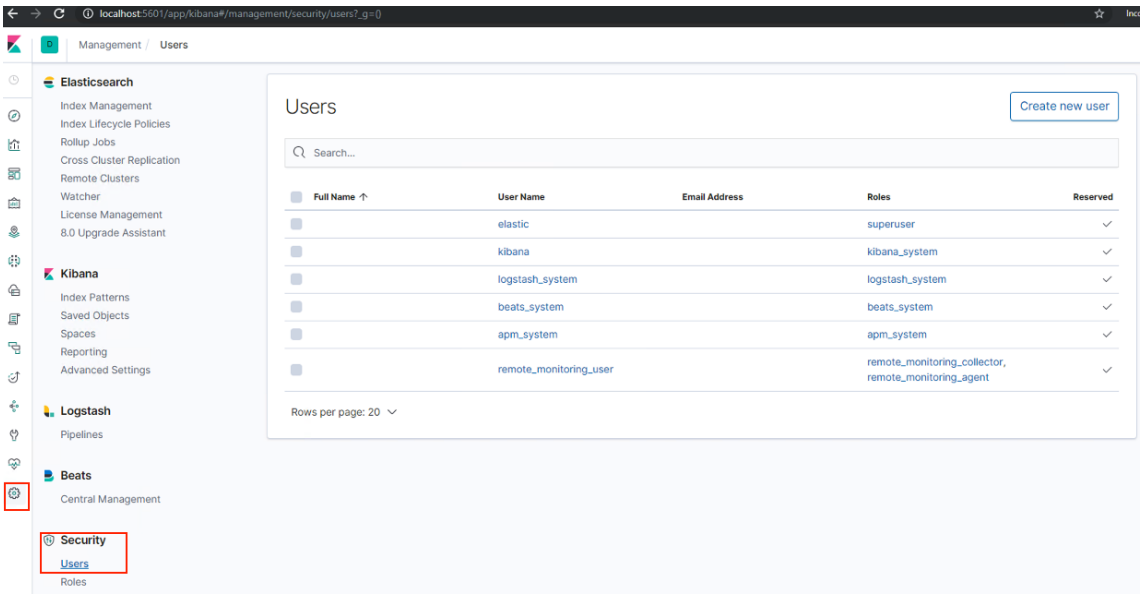
To check whether the import was successful, execute the following command and validate the count of documents:

```
curl -s -H "Content-Type: application/json" -u elastic:elastic -XGET
http://localhost:9200/employee,department/_count
{"count":6,"_shards":{"total":10,"successful":10,"skipped":0,"failed":0}}
```

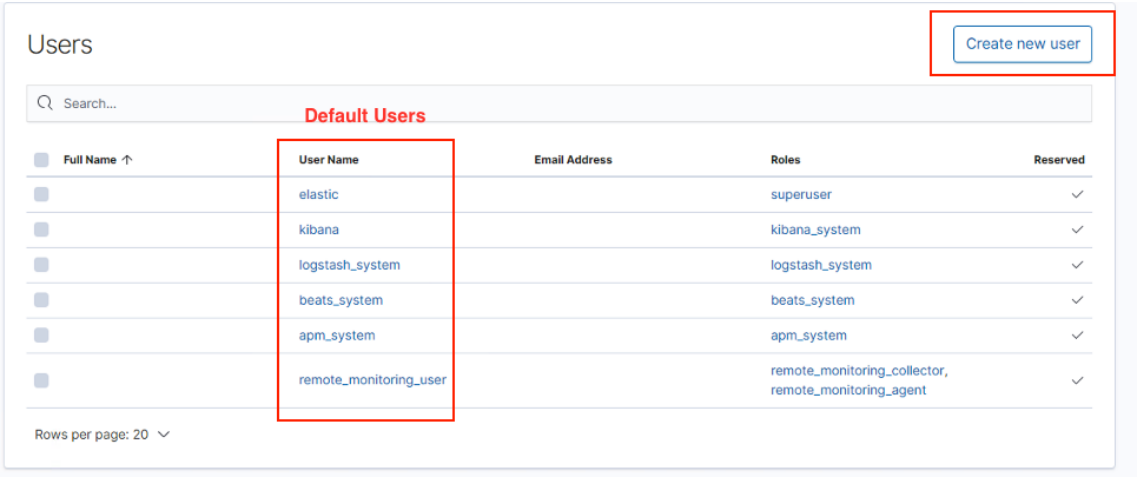
Creating a new user

Let's explore the creation of a new user in this section. Log in to Kibana (`http://locahost:5601`) as the `elastic` user:

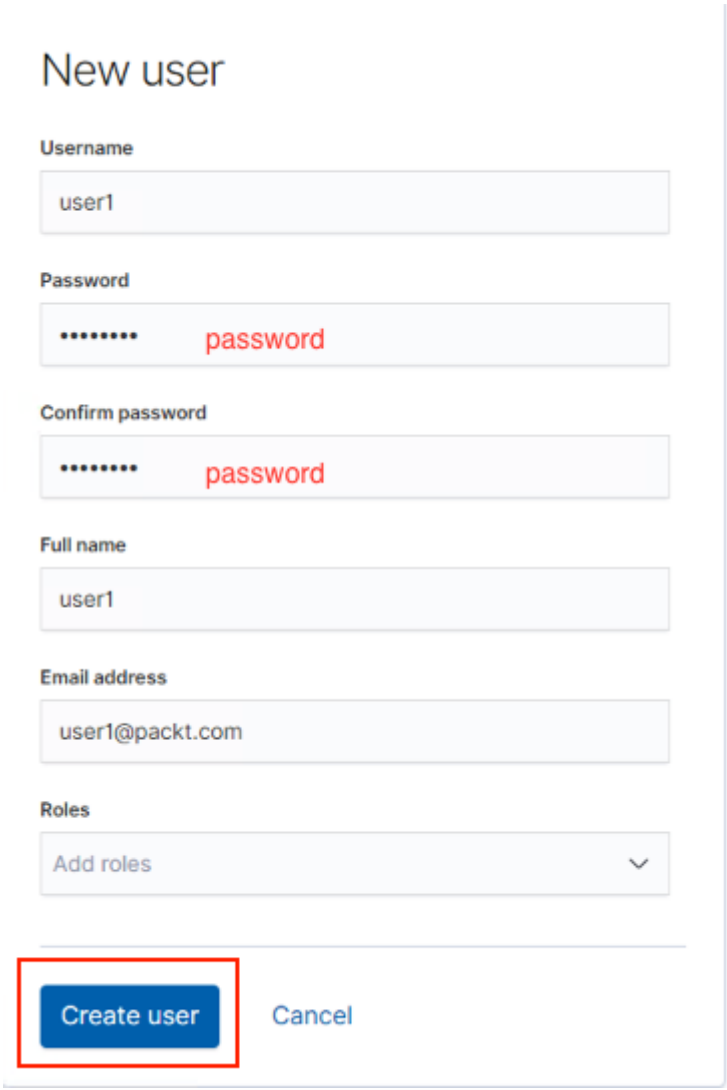
1. To create a new user, navigate to the `Management` UI and select `Users` in the `Security` section:



2. The `Users` screen displays the available users and their roles. By default, it displays the default/reserved users that are part of the native X-Pack security realm:



3. To create a new user, click on the `Create new user` button and enter the required details, as shown in the following screenshot. Then, click on `Create user` :



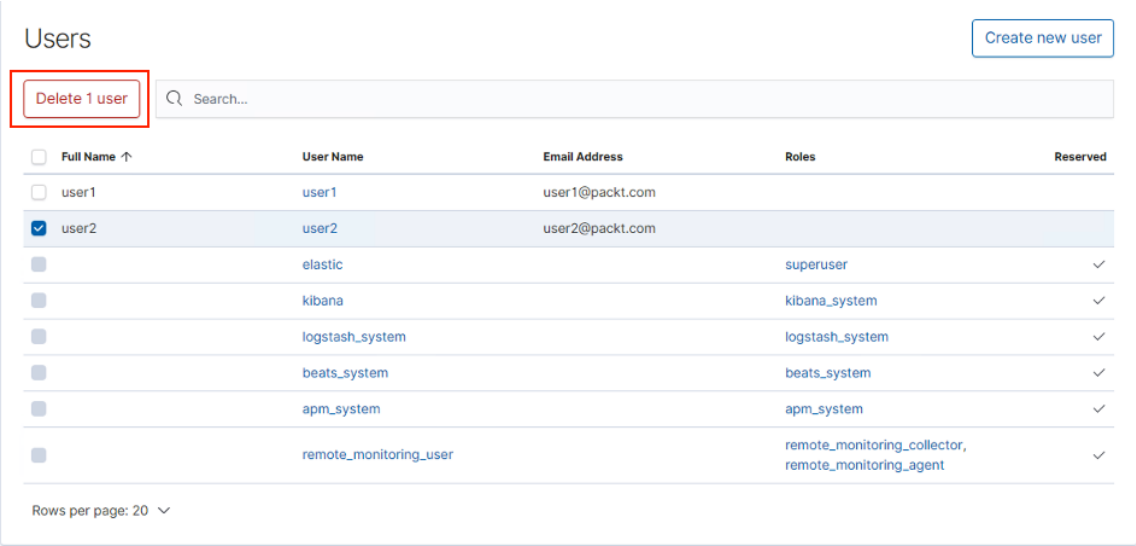
Now that the user has been created, let's try to access some Elasticsearch REST APIs with the new user credentials and see what happens. Execute the following command and check the response that's returned. Since the user doesn't have any role associated with them, the authentication is successful. The user gets HTTP status code `403`, stating that the user is not authorized to carry out the operation:

```
curl -s -H "Content-Type: application/json" -u user1:password -XGET
http://localhost:9200
Response:
{"error":{"root_cause":[{"type":"security_exception","reason":"action
[cluster:monitor/main] is unauthorized for user
[user1]"}],"type":"security_exception","reason":"action [cluster:monitor/main] is
unauthorized for user [user1]"},"status":403}
```

4. Similarly, go ahead and create one more user called `user2` with the password set as `password`.

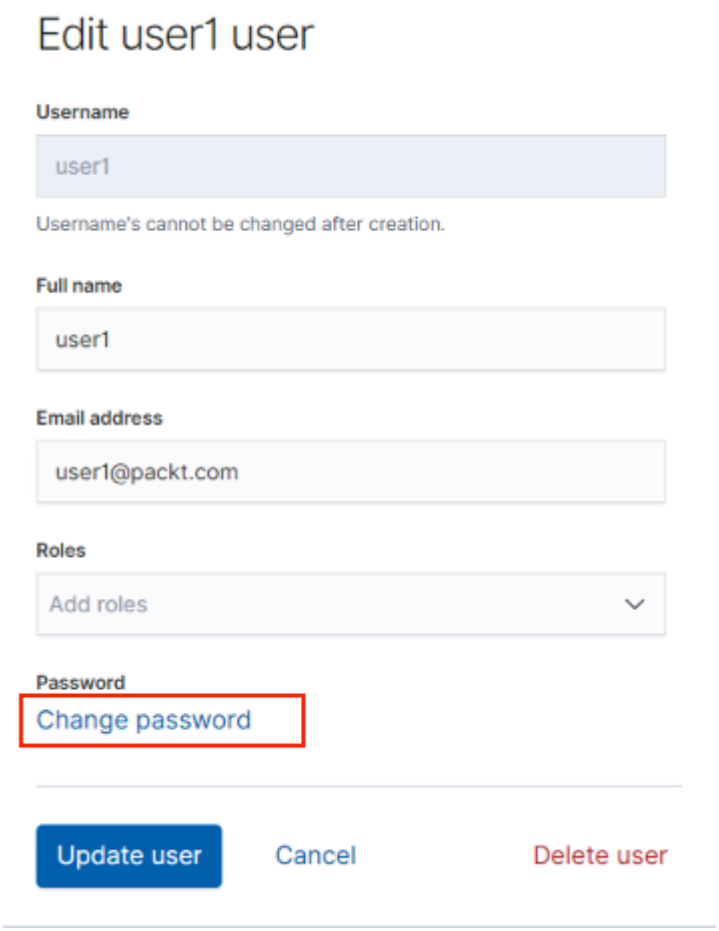
Deleting a user

To delete a role, navigate to `Users` UI, select the custom `user2` that you created, and click on the **Delete** button. You cannot delete built-in users:



Changing the password

Navigate to the `Users` UI and select the custom user for which the password needs to be changed. This will take you to the `User Details` page. You can edit the user's details, change their password, or delete the user from the user details screen. To change the user's password, click on the `Change password` link and enter the new password details. Then, click on the `Update user` button:

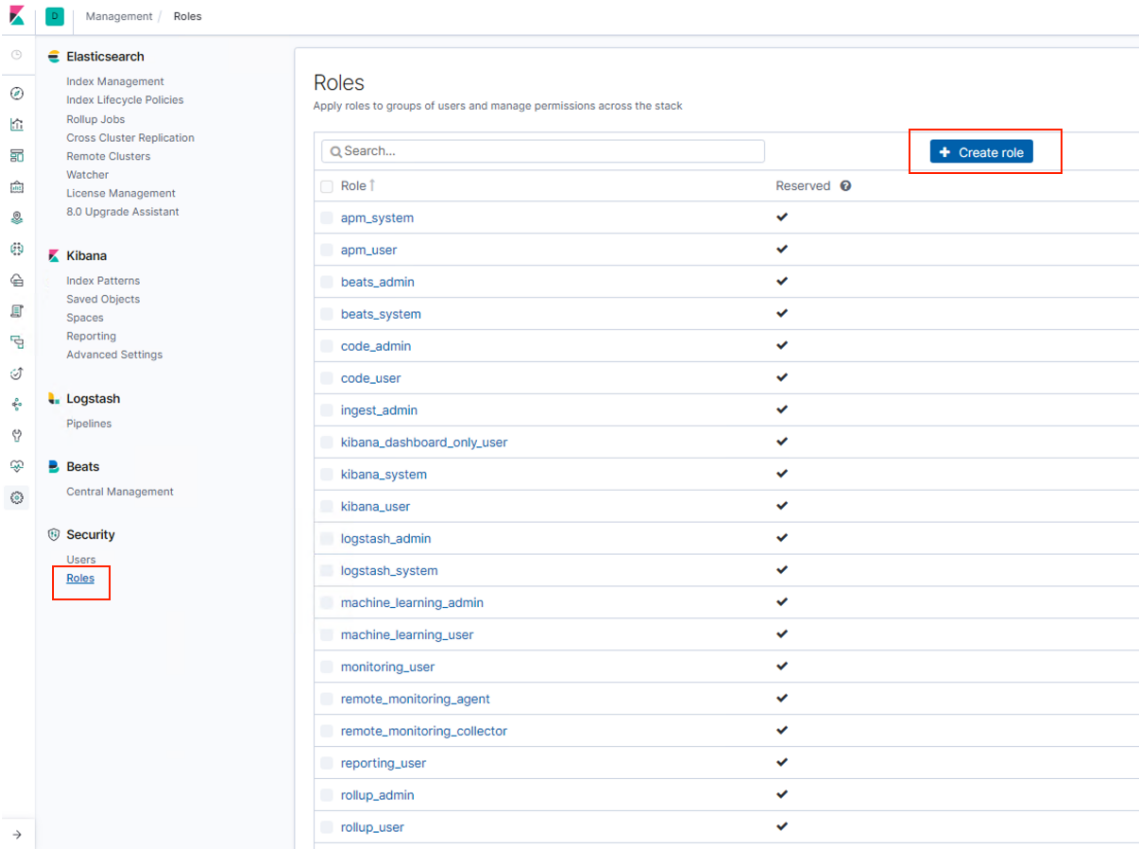


Note

The passwords must be a minimum of 6 characters long.

Creating a new role

To create a new user, navigate to the `Management` UI and select `Roles` in the `Security` section, or if you are currently on the `Users` screen, click on the `Roles` tab. The `Roles` screen displays all the roles that are defined/available. By default, it displays the built-in/reserved roles that are part of the X-Pack security native realm:



X-Pack security also provides a set of built-in roles that can be assigned to users. These roles are reserved and the privileges associated with these roles cannot be updated. Some of the built-in roles are as follows:

- `kibana_system` : This role grants the necessary access to read from and write to Kibana indexes, manage index templates, and check the availability of the Elasticsearch cluster. This role also grants read access for monitoring (`.monitoring-*`) and read-write access to reporting (`.reporting-*`) indexes. The default user, `kibana` , has these privileges.
- `superuser` : This role grants access for performing all operations on clusters, indexes, and data. This role also grants rights to create/modify users or roles. The default user, `elastic` , has superuser privileges.
- `ingest_admin` : This role grants permissions so that you can manage all pipeline configurations and all index templates.

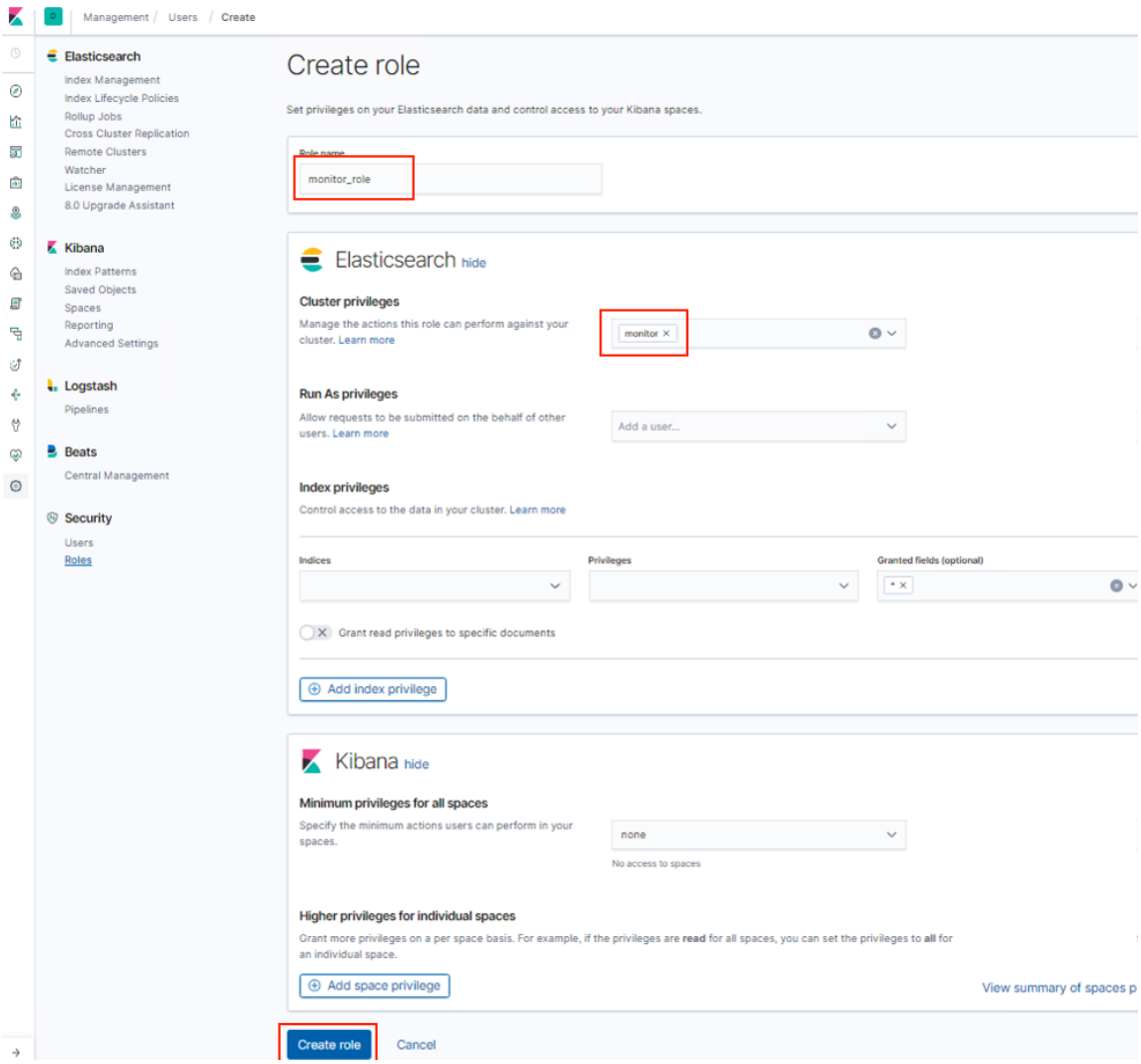
Note

To find the complete list of built-in roles and their descriptions, please refer to https://www.elastic.co/guide/en/x-pack/master/built-in-roles.html.

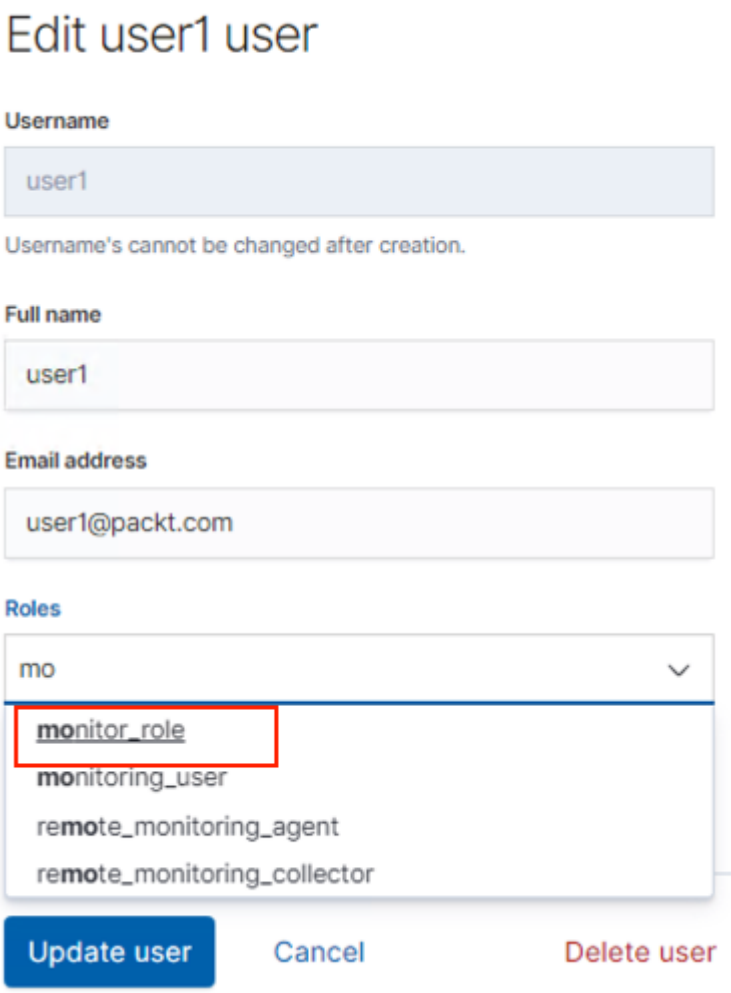
Users with the superuser role can create custom roles and assign them to the users using the Kibana UI.

Let's create a new role with a **Cluster privilege** called `monitor` and assign it to `user1` so that the user can cluster read-only operations such as cluster state, cluster health, nodes info, nodes stats, and more.

Click on the `Create role` button in the `Roles` page/tab and fill in the details that are shown in the following screenshot:



To assign the newly created role to `user1`, click on the `Users` tab and select `user1`. In the `User Details` page, from the roles dropdown, select the `monitor_role` role and click on the `Save` button, as shown in the following screenshot:



Note

A user can be assigned multiple roles.

Now, let's validate that `user1` can access some cluster/node details APIs:

```
curl -u user1:password "http://localhost:9200/_cluster/health?pretty"
{
  "cluster_name" : "elasticsearch",
  "status" : "yellow",
  "timed_out" : false,
  "number_of_nodes" : 1,
  "number_of_data_nodes" : 1,
  "active_primary_shards" : 5,
  "active_shards" : 5,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 2,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" : 0,
```

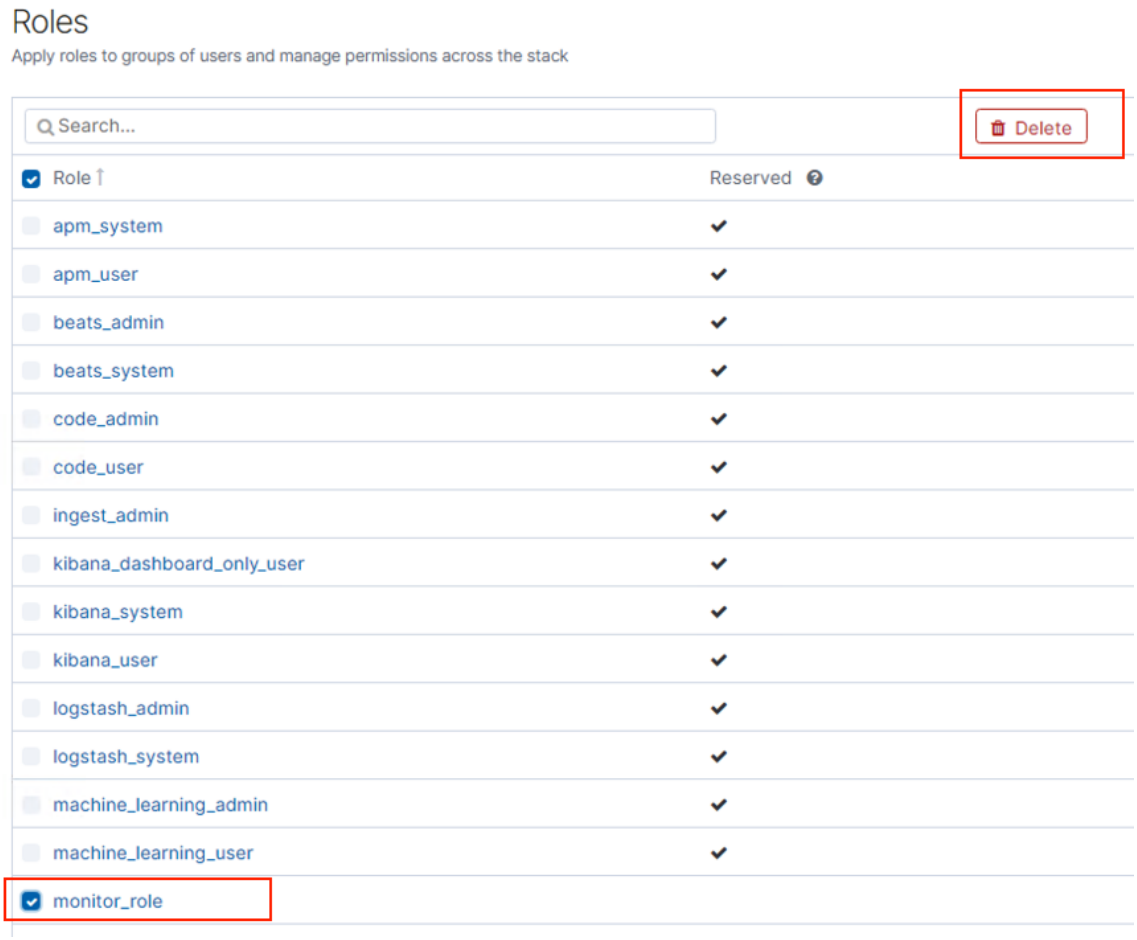
```
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 71.42857142857143
}
```

Let's also execute the same command that we executed when we created `user2`, but without assigning any roles to it, and see the difference:

```
curl -u user2:password "http://localhost:9200/_cluster/health?pretty"
{
  "error" : {
    "root_cause" : [
      {
        "type" : "security_exception",
        "reason" : "action [cluster:monitor/main] is unauthorized for user [user2]"
      }
    ],
    "type" : "security_exception",
    "reason" : "action [cluster:monitor/main] is unauthorized for user [user2]"
  },
  "status" : 403
}
```

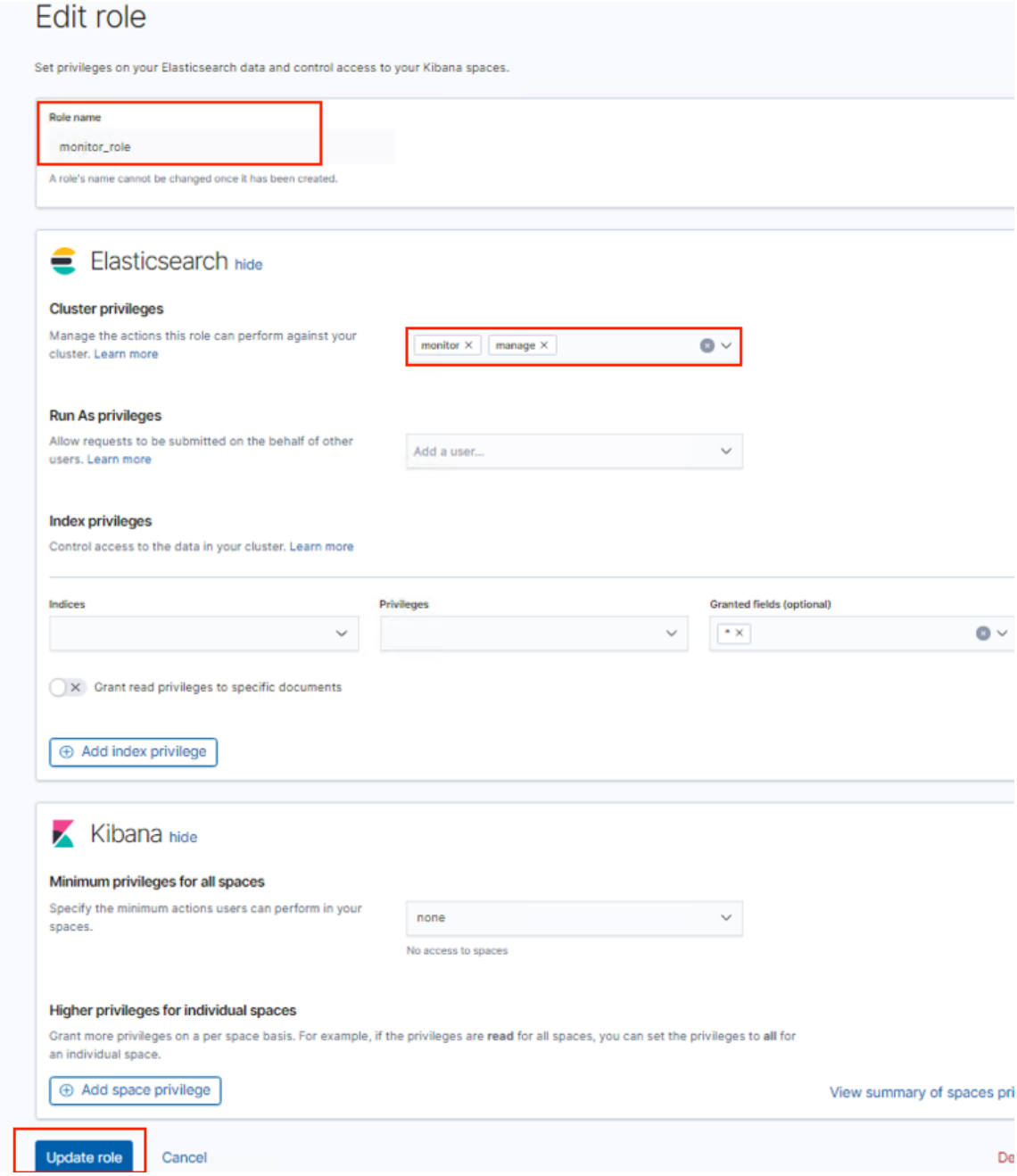
Deleting or editing a role

To delete a role, navigate to the `Roles` UI/tab, select the custom role that we created, and click on **Delete**. You cannot delete built-in roles:



To edit a role, navigate to the `Roles` UI/tab and click on the custom role that needs to be edited. The user is taken to the `Roles Details` page. Make the required changes in the `Privileges` section and click on the `Update role` button.

You can also delete the role from this page:



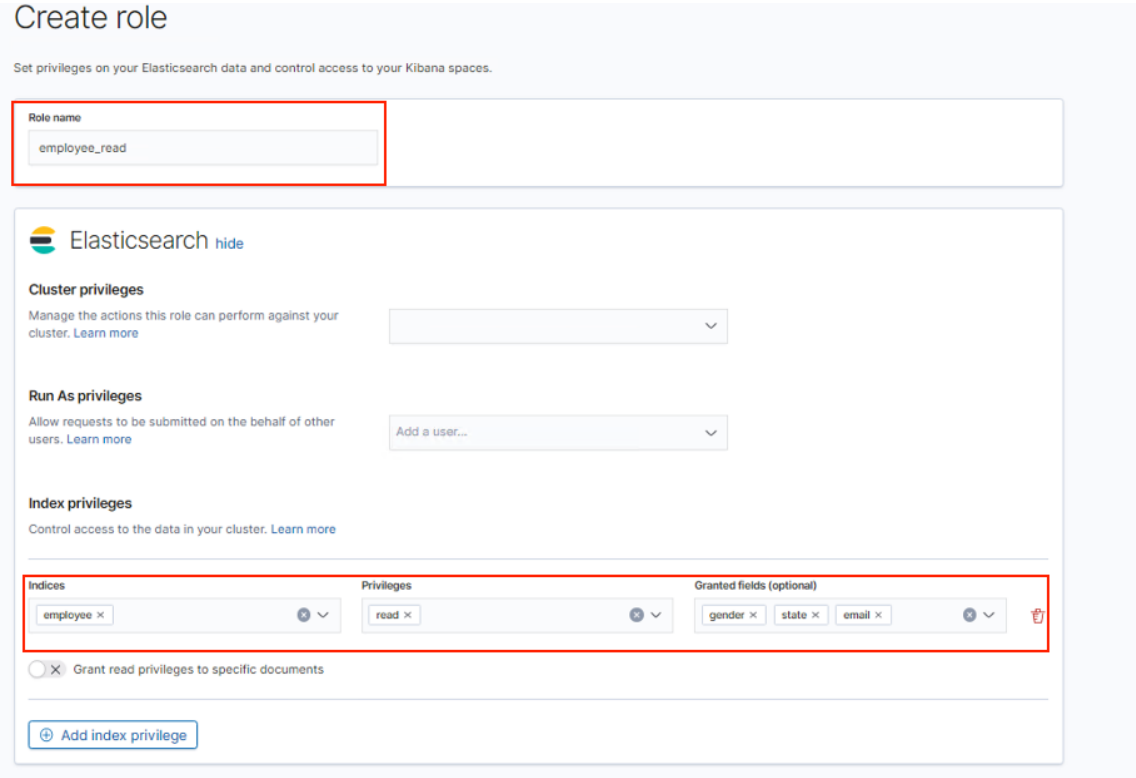
Document-level security or field-level security

Now that we know how to create a new user, create a new role, and assign roles to a user, let's explore how security can be imposed on documents and fields for a given index/document.

The sample data that we imported previously, at the beginning of this lab, contained two indexes: `employee` and `department`. Let's use these indexes and understand the document-level security with two use cases.

Use case 1: When a user searches for employee details, the user should not be able to find the salary/address details contained in the documents belonging to the `employee` index.

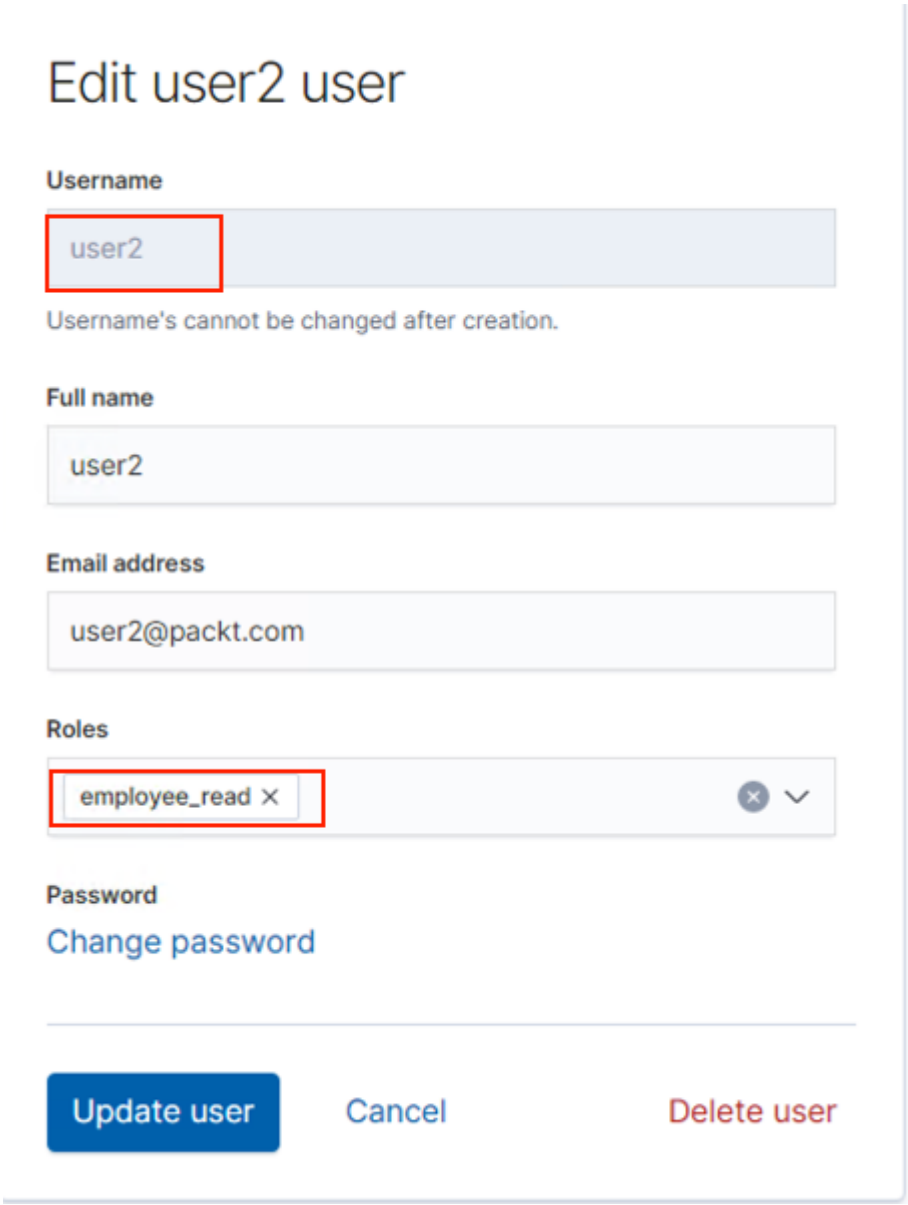
This is where field-level security helps. Let's create a new role (`employee_read`) with `read` index privileges on the `employee` index. To restrict the fields, type the actual field names that are allowed to be accessed by the user in the `Granted Fields` section, as shown in the following screenshot, and click the `Create role` button:



Note

When creating a role, you can specify the same set of privileges on multiple indexes by adding one or more index names to the `Indices` field, or you can specify different privileges for different indexes by clicking on the **Add index privilege** button that's found in the `Index privileges` section.

Assign the newly created role to `user2` :



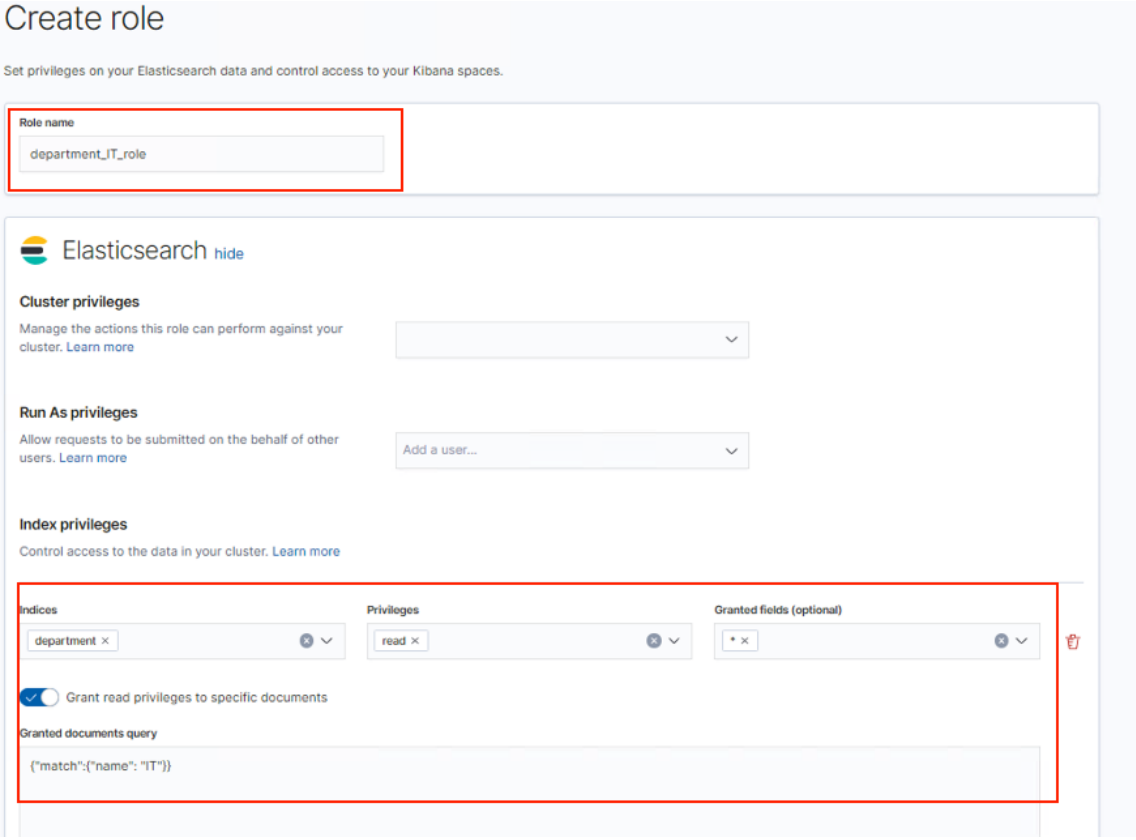
Now, let's search in the employee index and check what fields were returned in the response. As we can see in the following response, we have successfully restricted the user from accessing salary and address details:

```
curl -u user2:password "http://localhost:9200/employee/_search?pretty"
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
```

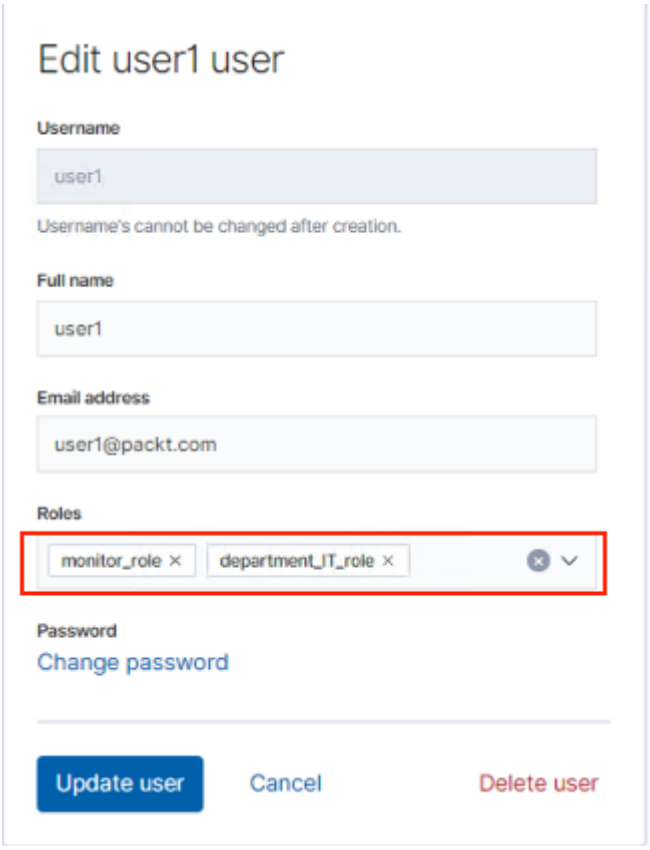
```
  "hits" : {
    "total" : {
      "value" : 3,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "employee",
        "_type" : "_doc",
        "_id" : "xsTc2GoBlyaBuhcfU42x",
        "_score" : 1.0,
        "_source" : {
"gender" : "M",
          "state" : "NE",
          "email" : "user1@fenago.com"
        }
      },
      {
        "_index" : "employee",
        "_type" : "_doc",
        "_id" : "x8Tc2GoBlyaBuhcfU42x",
        "_score" : 1.0,
        "_source" : {
 "gender" : "F",
          "state" : "OR",
          "email" : "user2@fenago.com"
        }
      },
      {
        "_index" : "employee",
        "_type" : "_doc",
        "_id" : "yMTc2GoBlyaBuhcfU42x",
        "_score" : 1.0,
        "_source" : {
 "gender" : "F",
          "state" : "CA",
          "email" : "user3@fenago.com"
        }
      }
    ]
  }
}
```


Use case 2: We want to have a multi-tenant index and restrict certain documents to certain users. For example, `user1` should be able to search in the department index and retrieve only documents belonging to the `IT` department.

Let's create a role, `department_IT_role`, and provide the `read` privilege for the `department` index. To restrict the documents, specify the query in the `Granted Documents Query` section. The query should be in the **Elasticsearch Query DSL** format:



Associate the newly created role with `user1`:



Let's verify that it is working as expected by executing a search against the `department` index using the `user1` credentials:

```
curl -u user1:password "http://localhost:9200/department/_search?pretty"
{
  "took" : 19,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 1,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "department",
        "_type" : "department",
        "_id" : "ycTc2GoBlyaBuhcfU42x",
        "_score" : 1.0,
```

```
        "_source" : {
"name" : "IT",
          "employees" : 50
        }
      }
    ]
  }
```

X-Pack security APIs

In the previous section, we learned how to manage users and roles using the Kibana UI. However, often, we would like to carry out these operations programmatically from our applications. This is where the X-Pack security APIs come in handy. X-Pack security APIs are REST APIs that can be used for user/role management, role mapping to users, performing authentication, and checking whether the authenticated user has specified a list of privileges. These APIs perform operations on the `native` realm. The Kibana UI internally makes use of these APIs for user/role management. In order to execute these APIs, the user should have `superuser` or the latest `manage_security` privileges. Let's explore some of these APIs in this section.

User Management APIs

This provides a set of APIs to create, update, or delete users from the `native` realm.

The following is a list of available APIs and how to use them:

```
GET /_xpack/security/user                                -- To list all the users
GET /_xpack/security/user/<username>                     -- To get the details of a
specific user
DELETE /_xpack/security/user/<username>                  -- To Delete a user
POST /_xpack/security/user/<username>                    -- To Create a new user
PUT /_xpack/security/user/<username>                     -- To Update an existing user
PUT /_xpack/security/user/<username>/_disable            -- To disable an existing user
PUT /_xpack/security/user/<username>/_enable             -- To enable an existing
disabled user
PUT /_xpack/security/user/<username>/_password           -- to Change the password
```

The `username` in the path parameter specifies the user against which the operation is carried out. The body of the request accepts parameters such as `email`, `full_name`, and `password` as strings and `roles` as list.

Example 1: Create a new user, `user3`, with `monitor_role` assigned to it:

```
curl -u elastic:elastic -X POST http://localhost:9200/_xpack/security/user/user3 -H
'content-type: application/json' -d '
{
  "password" : "randompassword",
  "roles" : [ "monitor_role"],
  "full_name" : "user3",
  "email" : "user3@fenago.com"

}'

Response:
user":{"created":true}}
```

Example 2: Get the list of all users:

```
curl -u elastic:elastic -XGET http://localhost:9200/_xpack/security/user?pretty
```

Example 3: Delete `user3`:

```
curl -u elastic:elastic -XDELETE http://localhost:9200/_xpack/security/user/user3
Response:
{"found":true}
```

Example 4: Change the password:

```
curl -u elastic:elastic -XPUT
http://localhost:9200/_xpack/security/user/user2/_password -H "content-type:
application/json" -d "{ \"password\": \"newpassword\"}"
```

Note

When using `curl` commands on Windows machines, note that they don't work if they have single quotes (') in them. The preceding example showed the use of a `curl` command on a Windows machine. Also, make sure you escape double quotes within the body of the command, as shown in the preceding example.

Role Management APIs

This provides a set of APIs to create, update, remove, and retrieve roles from the `native` realm.

The list of available APIs under this section, as well as information on what they do, is as follows:

```
GET /_xpack/security/role                                  -- To retrieve the list of all
roles
GET /_xpack/security/role/<rolename>                           -- To retrieve details of
a specific role
POST /_xpack/security/role/<rolename>/_clear_cache             -- To evict/clear roles
from the native role cache
POST /_xpack/security/role/<rolename>                          -- To create a role
PUT /_xpack/security/role/<rolename>                           -- To update an existing
role
```

The `rolename` in the path parameter specifies the role against which the operation is carried out. The body of the request accepts parameters such as `cluster`, which accepts a list of cluster privileges; `indices`, which accepts a list of objects that specify the indices privileges and `run_as`, which contains a list of users that the owners of this role can impersonate.

`indices` contains an object with parameters such as `names`, which accepts a list of index names; `field_security`, which accepts a list of fields to provide read access; `privileges`, which accepts a list of index privileges; and the `query` parameter, which accepts the query to filter the documents.

Let's take a look at a few examples of managing different roles using APIs:

- **Example 1**: Create a new role with field-level security imposed on the employee index:

```
curl -u elastic:elastic -X POST
http://localhost:9200/_xpack/security/role/employee_read_new -H 'content-type:
application/json' -d '{

  "indices": [
```

```
    {
      "names": [ "employee" ],
      "privileges": [ "read" ],
      "field_security" : {
        "grant" : [ "*" ],
        "except": [ "address*","salary" ]
      }

    }
  ]
}'

Response:
role":{"created":true}}
```

Note

Unlike the Kibana UI, which doesn't have any way to exclude fields from user access, using the Security API, you can easily exclude or include fields as part of field-level security. In the preceding example, we have restricted access to the `salary` field and any fields starting with the `address` text/string.

- **Example 2**: Get the details of a specific role:

```
curl -u elastic:elastic -XGET
http://localhost:9200/_xpack/security/role/employee_read_new?pretty
Response:
{
  "employee_read" : {
    "cluster" : [ ],
    "indices" : [
      {
        "names" : [
          "employee"
        ],
        "privileges" : [
          "read"
        ],
        "field_security" : {
          "grant" : [
            "*"
          ],
          "except" : [
            "address*",
            "salary"
          ]
        }
      }
    ],
    "run_as" : [ ],
    "metadata" : { },
    "transient_metadata" : {
      "enabled" : true
    }
```

```
  }
}
```

- **Example 3**: Delete a role:

```
curl -u elastic:elastic -XDELETE
http://localhost:9200/_xpack/security/role/employee_read

Response:
{"found":true}
```

Note

Similar to the User Management and Role Management APIs, using Role Mapping APIs, you can associate roles with users. Details about Role Mapping APIs and User Management APIs can be found at https://www.elastic.co/guide/en/elasticsearch/reference/master/security-api-role-mapping.html and https://www.elastic.co/guide/en/elasticsearch/reference/master/security-api-users.html, respectively.

Monitoring Elasticsearch

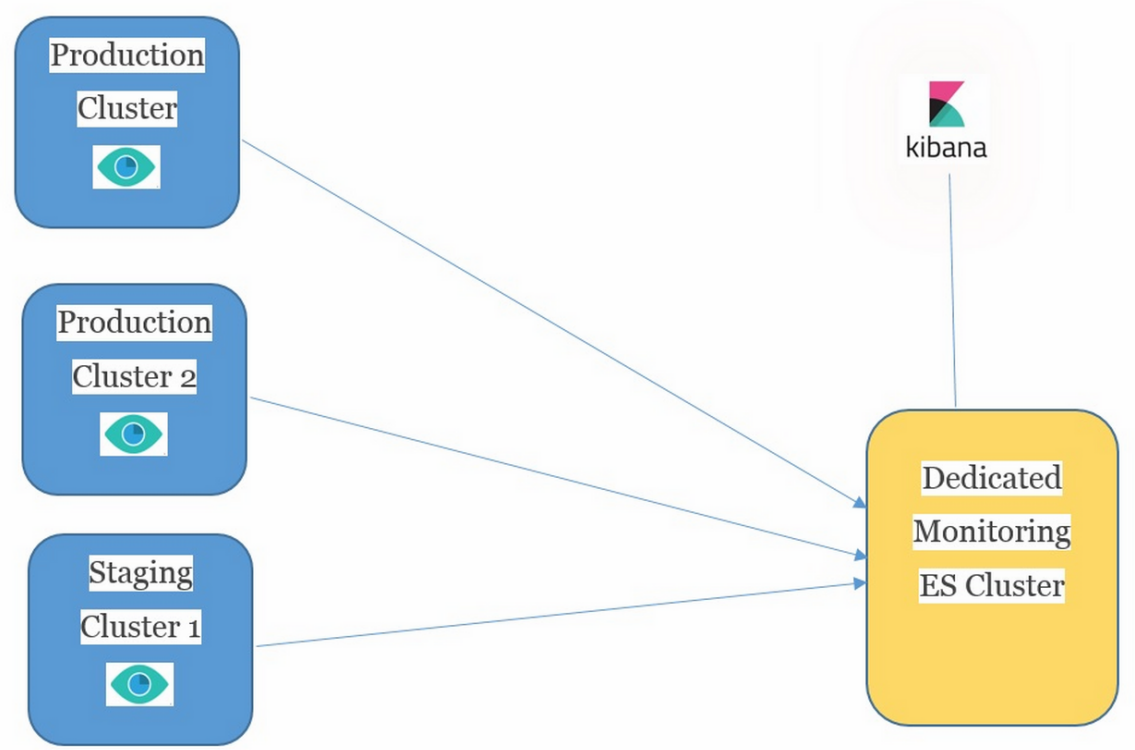
Elasticsearch exposes a rich set of APIs, known as **stats** APIs, to monitor Elasticsearch at the cluster, node, and indices levels. Some of these APIs are `_cluster/stats`, `_nodes/stats`, and `myindex/stats`. These APIs provide state/monitoring information in real time, and the statistics that are presented in these APIs are point-in-time and in `.json` format. As an administrator/developer, when working with Elasticsearch, you will be interested in both real-time statistics as well as historical statistics, which would help you in understanding/analyzing the behavior (health or performance) of a cluster better.

Also, reading through a set of numbers for a period of time (say, for example, to find out the JVM utilization over time) would be very difficult. Rather, a UI that pictorially represents these numbers as graphs would be very useful for visualizing and analyzing the current and past trends/behaviors (health or performance) of the Elasticsearch cluster. This is where the monitoring feature of X-Pack comes in handy.

The X-Pack monitoring components allow you to easily monitor the Elastic Stack (Elasticsearch, Kibana, and Logstash) from Kibana. X-Pack consists of a monitoring agent that runs on each of the instances (Elasticsearch, Kibana, and Logstash) and periodically collects and indexes the health and performance metrics. These can then be easily visualized using the Monitoring UI component of Kibana. The Monitoring UI of Kibana comes with predefined dashboards that let you easily visualize and analyze real-time and past performance data.

By default, the metrics collected by X-Pack are indexed within the cluster you are monitoring. However, in production, it is strongly recommended to have a separate, dedicated cluster to store these metrics. A dedicated cluster for monitoring has the following benefits:

- Allows you to monitor multiple clusters from a central location
- Reduces the load and storage on your production clusters since the metrics are stored in a dedicated monitoring cluster
- There is access to **Monitoring**, even when some clusters are unhealthy or down
- Separate security levels from **Monitoring** and **Production Cluster** can be enforced:



As we mentioned previously, the metrics collected by X-Pack are indexed within the cluster you are monitoring. If a dedicated monitoring cluster is set up, then we need to configure where to send/ship the metrics to in the monitored instances. This can be configured in the `elasticsearch.yml` file of each node, as shown in the following code:

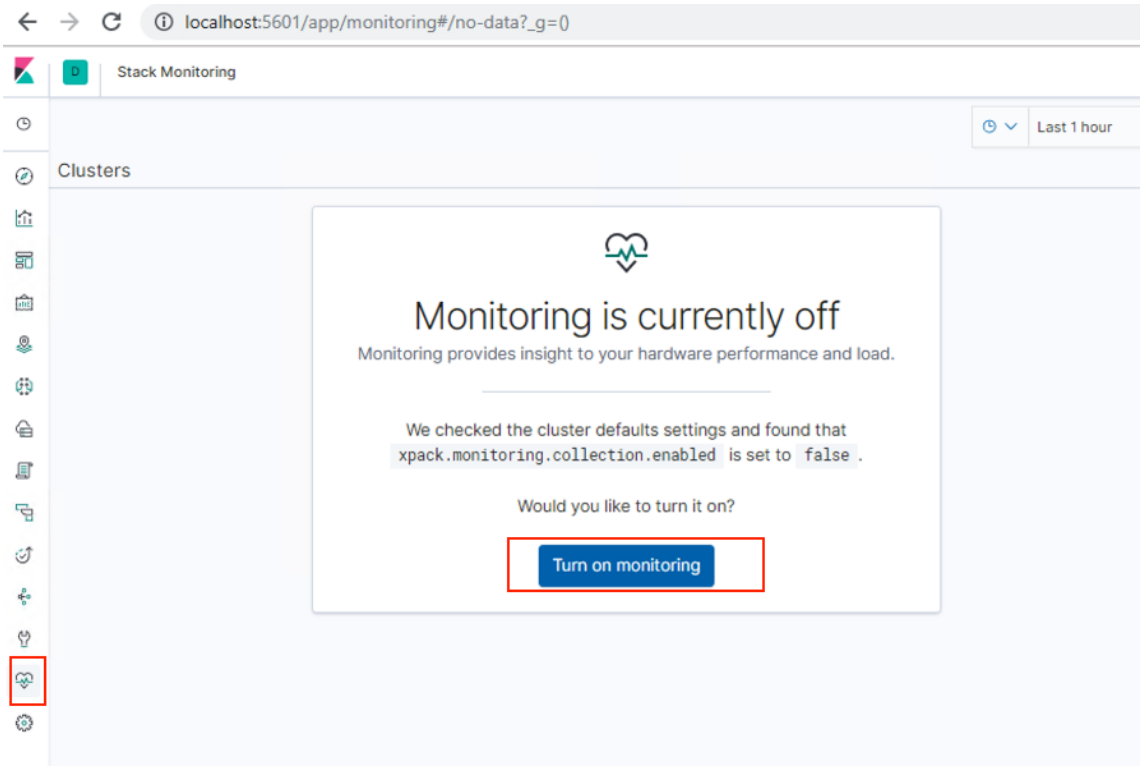
```
xpack.monitoring.exporters:
    id1:
    type: http
    host: ["http://dedicated_monitoring_cluster:port"]
```

Note

It's optional to have X-Pack installed on a dedicated monitoring cluster; however, it is recommended to have it installed there too. If X-Pack is installed on a dedicated monitoring cluster, then make sure you provide the user credentials (`auth.username` and `auth.password`) as well while configuring the monitored instances. Monitored metrics are stored in a system-level index that has the `.monitoring-*` index pattern.

Monitoring UI

To access the Monitoring UI, log in to Kibana and click on Stack Monitoring from the side navigation. If the monitoring data collection is not enabled, you will be taken to the following screen, where you can enable monitoring by clicking on the `Turn on monitoring` button. By default, monitoring would be enabled but data collection would be disabled. These settings can be dynamic and can be updated using the **cluster update settings** API, which doesn't require a restart to occur. If the same settings were set in `elasticsearch.yml` or `kibana.yml` , a restart would be required:



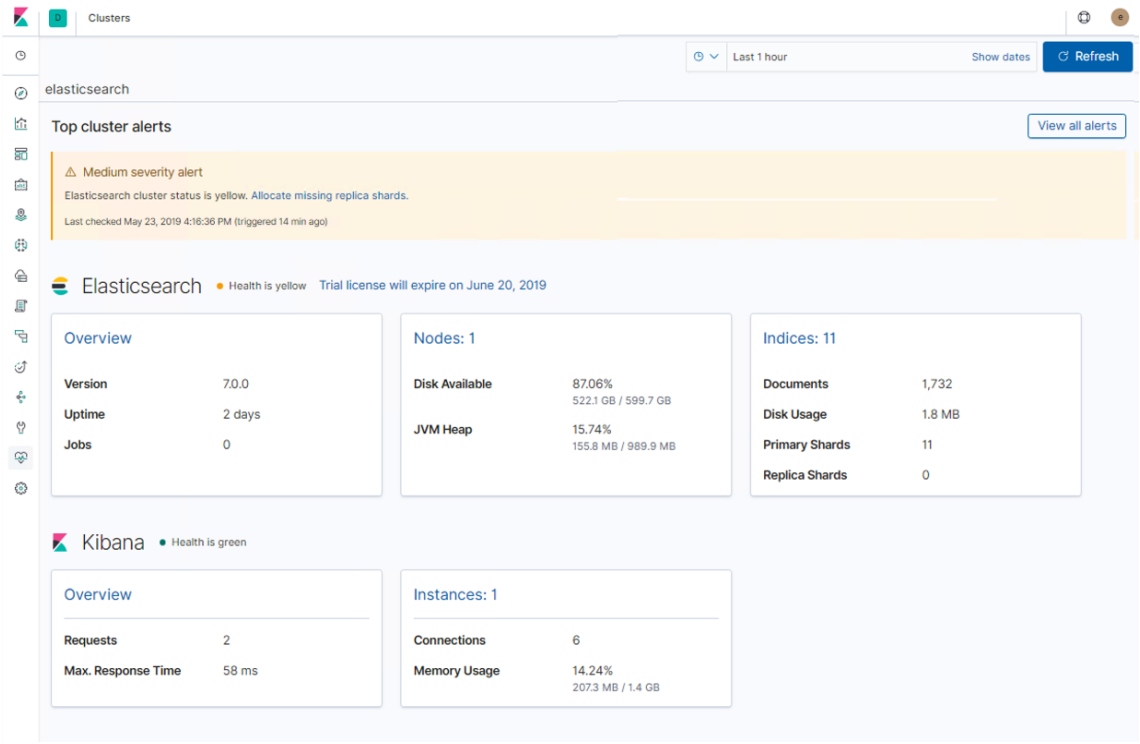
Once you click on `Turn on monitoring`, the cluster settings will update, which can be verified by using the following API:

```
curl -u elastic:elastic -XGET http://localhost:9200/_cluster/settings?pretty
{
  "persistent" : {
"xpack" : {
      "monitoring" : {
        "collection" : {
          "enabled" : "true"
        }
      }
    }
  },
  "transient" : { }
}
```

Note

You can refer to https://www.elastic.co/guide/en/elasticsearch/reference/7.0/monitoring-settings.html and https://www.elastic.co/guide/en/kibana/7.0/monitoring-settings-kb.html for more on how to customize the monitoring settings.

Once data collection has been enabled, click on the Stack Monitoring icon on the left-hand side menu. You will see the following screen:



This page provides a summary of the metrics that are available for Elasticsearch and Kibana. By clicking on links such as `Overview`, `Nodes`, `Indices`, or `Instances`, you can get additional/detailed information. The metrics that are displayed on the page are automatically refreshed every 10 seconds, and by default, you can view the data of the past 1 hour, which can be changed in the `Time Filter` that's found toward the top left of the screen. You can also see the cluster name, which in this case is `elasticsearch`.

Note

The monitoring agent that's installed on the instances being monitored sends metrics every 10 seconds by default. This can be changed in the configuration file (`elasticsearch.yml`) by setting the appropriate value to the `xpack.monitoring.collection.interval` property.

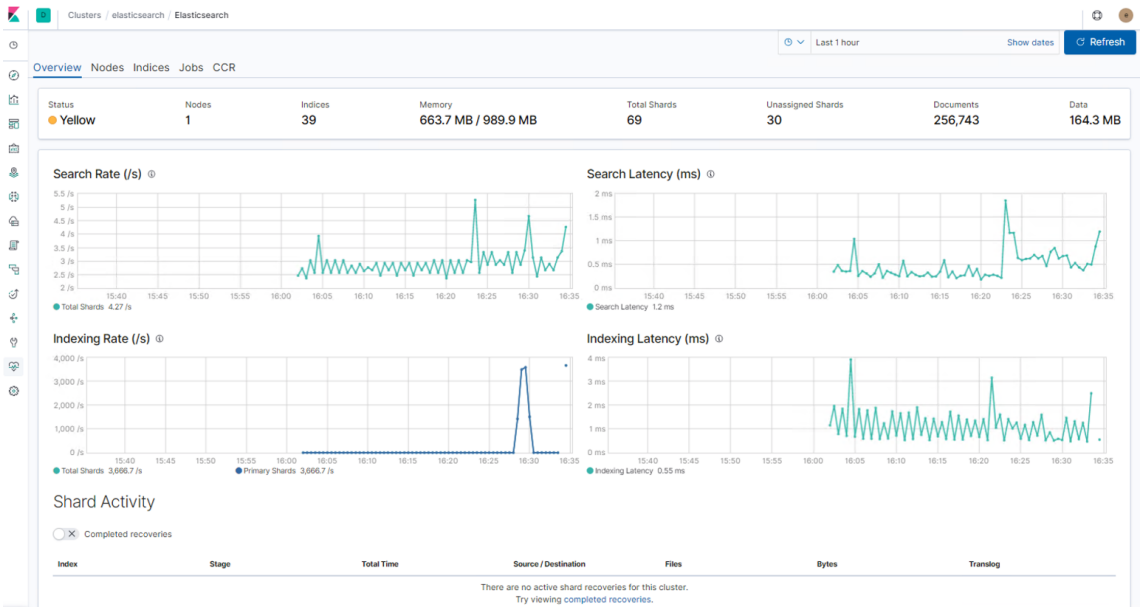
Elasticsearch metrics

You can monitor the Elasticsearch performance data at a cluster level, node level, or index level. The Elasticsearch Monitoring UI provides three tabs, each displaying the metrics at the cluster, node, and index levels. The three tabs are `Overview`, `Nodes`, and `Indices`, respectively. To navigate to the Elasticsearch Monitoring UI, click on one of the links (`Overview`, `Nodes`, and `Indices`) under the Elasticsearch section.

Overview tab

Cluster-level metrics provide aggregated information across all the nodes and is the first place one should look when monitoring an Elasticsearch cluster. Cluster-level metrics are displayed in the `Overview` tab and can be navigated to by clicking on the `Overview` link under the Elasticsearch section found in the landing page of the Monitoring UI.

The `Overview` tab provides key metrics that indicate the overall health of an Elasticsearch cluster:



The key metrics that are displayed are cluster status, number of nodes and number of indices present, memory used, total number of shards present, total number of unassigned shards, total number of documents present in the indices, the disk space used for storing the documents, uptime, and version of Elasticsearch. The `Overview` tab also displays charts that show the search and indexing performance over time, while the table at the bottom shows information about any shards that are being recovered.

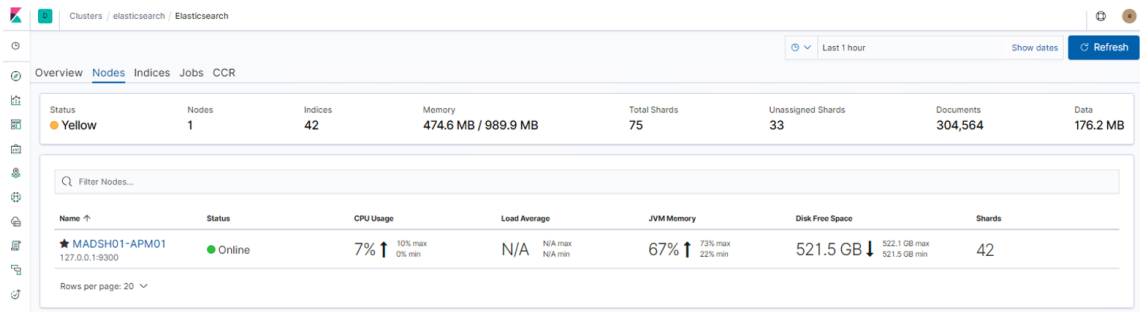
Note

Clicking on the Information icon present at the top right of each chart provides a description of the metrics.

In the `Overview` tab, the metrics are aggregated at the cluster level; so, when you're monitoring the Elasticsearch cluster, you might miss out some vital parameters that may eventually affect the cluster's overall state. For example, the `Memory Used` metric showcases the average memory used by combining the memory used across all nodes. However, one node might be running with full memory utilization and another node's memory might have hardly been used. Hence, as an administrator, you should always monitor at the `Node` level too.

Nodes tab

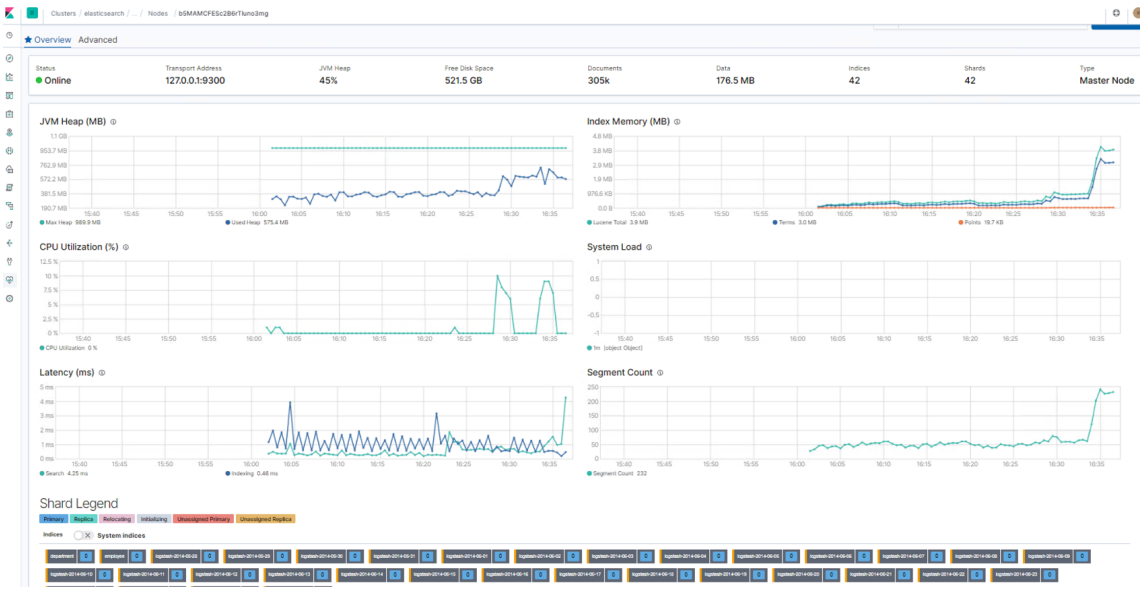
Clicking on the `Nodes` tab displays the summary details of each node present in the cluster, as shown in the following screenshot:



For each node, information is provided, such as the `Name` of the node, `Status` of the node, `CPU Usage` (average, min, and max usage), `Load Average` (average, min, and max usage), `JVM Memory` (average,

min, and max usage), `Disk Free Space` (average, min, and max usage), and total number of assigned `Shards`. It also provides information such as whether a node is a `Master node` or not (indicated by a star next to the node name) and details about the transport host and port.

Clicking on the `Node` name provides detailed information about the node. This detailed information is displayed in two tabs, namely `Overview` and `Advanced`. The node `Overview` tab looks like this:



The node `Overview` tab provides information in the top pane, such as the status of the node, transport IP address of the node, JVM Heap Utilization in percent, free disk space available, total number of documents present on the node (this number includes documents present in both replica and primary shards), total disk space used, total number of indices in the node, total number of shards, and type of node (master, data, ingest, or coordinating node).

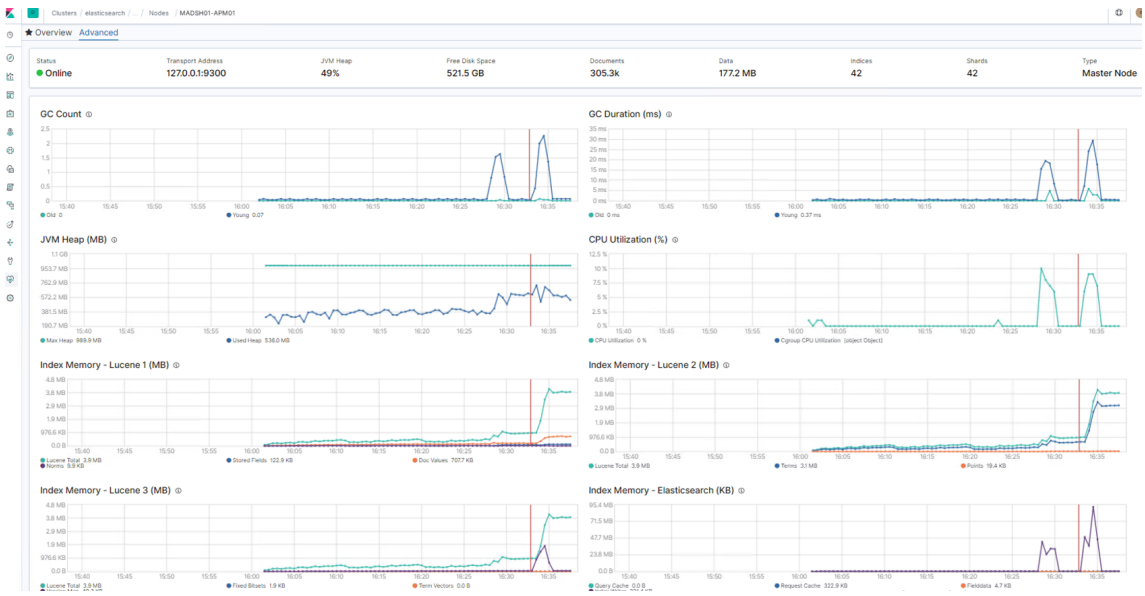
The node `Overview` tab also provides visualizations for `JVM Heap` usage, `Index Memory`, `CPU Utilization` in percent, `System Load` average, `Latency (ms)`, and `Segment Count`. The statuses of shards of various indices are provided under the `Shard Legend` section.

Note

If the `Show system indices` checkbox is checked, then the shard status of all the indexes created by X-Pack can be seen.

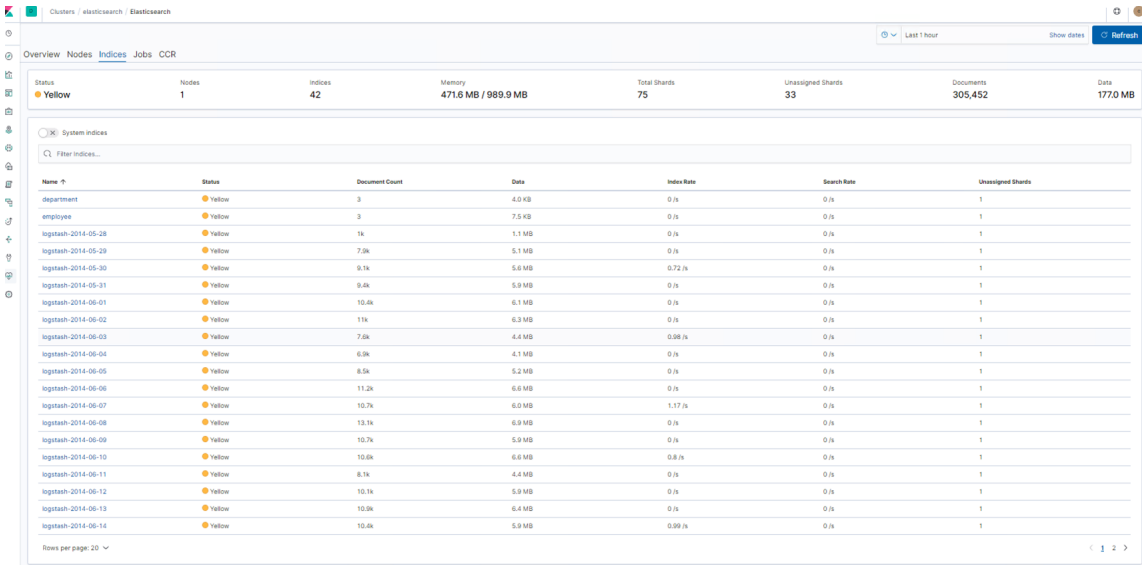
The node `Advanced` tab provides visualizations of other metrics, such as **garbage collection** (GC) count and duration, detailed Index Memory usage at Lucene and Elasticsearch levels, Indexing Time (in ms), Request rate, Indexing, Read Threads, and Cgroup stats.

The following is a screenshot of the Node `Advanced` tab:



The Indices tab

Clicking on the `Indices` tab displays the summary details of each index present in the cluster, as shown in the following screenshot:

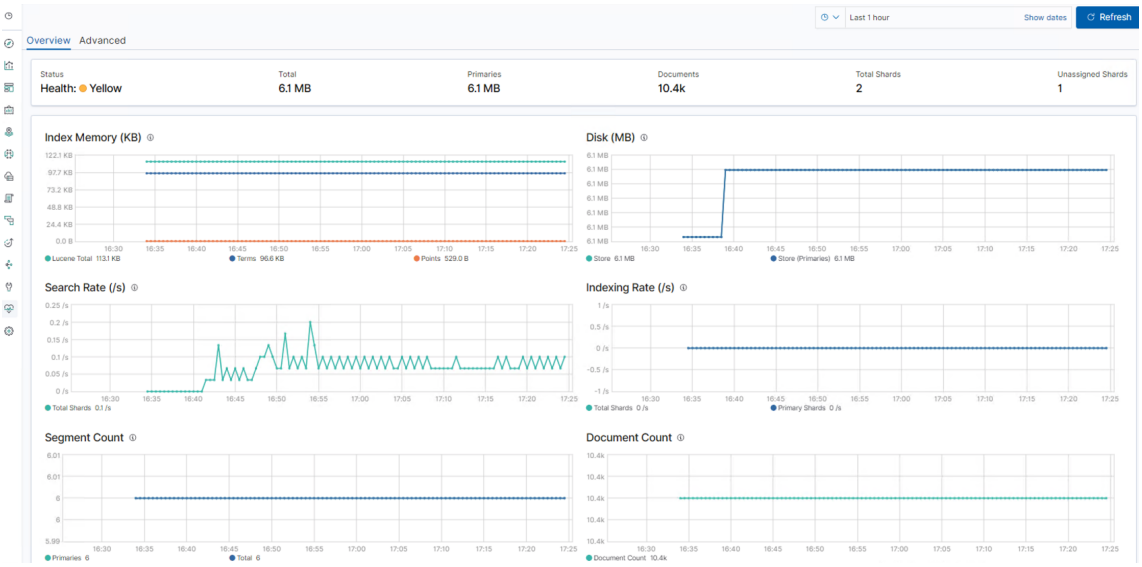


Note

If the `Show system indices` checkbox is checked, then the shard status of all indexes created by X-Pack can be seen.

For each index, it provides information, such as the name of the index, status of the index, total count of documents present, disk space used, index rate per second, search rate per second, and number of unassigned shards.

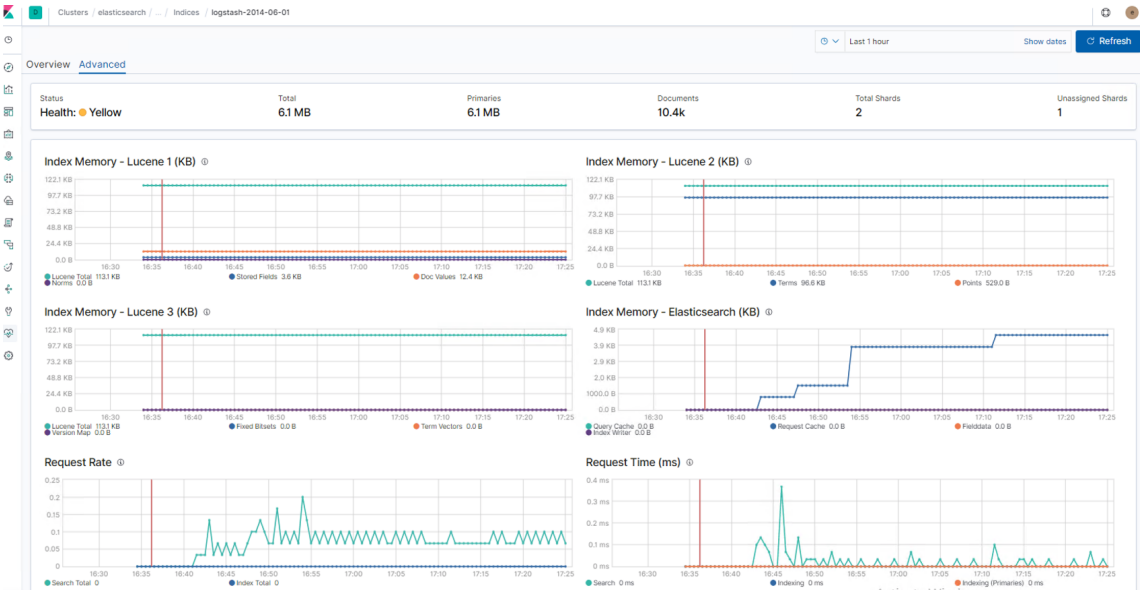
Clicking on an `Index` name provides detailed information about that index. The detailed information is displayed in two tabs, namely `Overview` and `Advanced`. The Index `Overview` tab looks like this:



This tab provides information in the top pane, such as on the status of the index, total number of documents present in the index, disk space used, total number of shards (primary and replicas), and unassigned shards.

The Index `Overview` tab also provides visualizations for Index Memory (in KB), Index size (in MB), Search rate per second, Indexing rate per second, total count of segments, and total count of documents. `Shard Legend` displays the status of shards belonging to the index and the information for the nodes the shards are assigned to.

The Index `Advanced` tab provides visualizations of other metrics, such as detailed **Index Memory** usage at Lucene and Elasticsearch levels, **Indexing Time** (in ms), **Request Rate** and **Time**, **Refresh Time** (in ms), **Disk usage**, and **Segment counts**:



Note

From the landing page of the `Monitoring` UI, by clicking on **Overview** or **Instances** under the Kibana section, the metrics of Kibana can be visualized/monitored in a similar way.

Alerting

The Kibana UI provides beautiful visualizations that help with analyzing and detecting anomalies in data in real time. However, as an administrator or an analyst, it wouldn't be possible to sit in front of dashboards for hours to detect anomalies and take action. It would be nice if the administrator gets notified when, for example, the following events occur:

- There is an outage in one of the servers being monitored
- An Elasticsearch Cluster turns red/yellow due to some nodes leaving the cluster
- Disk space/CPU utilization crosses a specific threshold
- There is an intrusion in the network
- There are errors reported in the logs

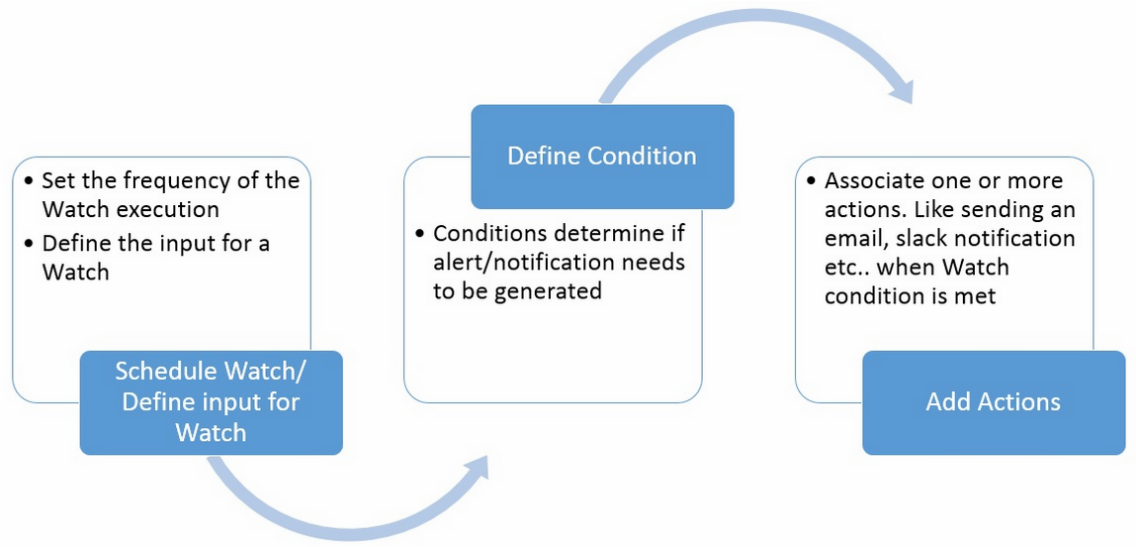
This is where the X-Pack Alerting component comes to the rescue. The X-Pack Alerting component, named `Watcher`, provides the ability to automatically watch for changes/anomalies in data stored on Elasticsearch and take the required action. X-Pack Alerting is enabled by default as part of the X-Pack default installation.

`Watcher` provides a set of REST APIs for creating, managing, and testing watches. Kibana also provides a **Watcher** UI for creating, managing, and testing. The Watcher UI internally makes use of Watcher REST APIs for the management of watches.

Anatomy of a watch

A **Watch** is made of the following components:

- `schedule` : This is used to specify the time interval for scheduling/triggering the watch.
- `query` : This is used to specify a query to retrieve data from Elasticsearch and run it as an input to the condition. Elasticsearch Query DSL/Lucene queries can be used to specify the queries.
- `condition` : This is used to specify conditions against the input data obtained from the query and check whether any action needs to be taken or not.
- `action` : This is used to specify actions such as sending an email, sending a slack notification, logging the event to a specific log, and much more on meeting the condition:



Let's look into a sample watch and understand the building blocks of a watch in detail. The following code snippet creates a watch:

```
curl -u elastic:elastic -X POST
http://localhost:9200/_xpack/watcher/watch/logstash_error_watch   -H 'content-type:
application/json'   -d '{
"trigger":{"schedule":{"interval":"30s"}},"input":{"search":{"request":{"indices":
["logstash*"],"body":{"query":{"match":{"message":"error"}}}}}},"condition":
{"compare":{"ctx.payload.hits.total":{"gt":0}}},"actions":{"log_error":{"logging":
{"text":"The number of errors in logs is  {{ctx.payload.hits.total}}"}}}}'
```

Note

In order to create a watch, the user should have `watcher_admin` cluster privileges.

- `trigger` : This section is used to provide a schedule to specify how often the watch needs to be executed. Once the watch is created, `Watcher` immediately registers its trigger with the `scheduler` trigger engine. The trigger engine evaluates the trigger and runs the watch accordingly.

Several types of schedule triggers can be defined to specify when watch execution should start. The different types of schedule triggers are interval, hourly, daily, weekly, monthly, yearly, and cron.

In the preceding code snippet, a trigger was specified with a schedule of 30 seconds, which means that the watch is executed every 30 seconds.

Example to specify hourly trigger: The following snippet shows how to specify an hourly trigger that triggers the watch every 45th minute of an hour:

```
{"trigger":{"schedule":{"hourly":{"minute":45}}}}
```

You can specify an array of minutes, too. The following snippet shows how to specify an hourly trigger that triggers the watch every 15th and 45th minute of an hour:

```
{"trigger":{"schedule":{"hourly":{"minute":[ 15,45]}}}}
```

The following is an example of specifying that the watch should trigger daily at 8 PM:

```
{"trigger":{"schedule":{"daily":{"at":"20:00"}}}}
```

The following is an example of specifying a watch to trigger weekly on Mondays at 10 AM and on Fridays at 8 PM:

```
{"trigger":{"schedule":{"weekly":[{"on":"monday","at":"10:00"},
{"on":"friday","at":"20:00"}]}}}
```

The following is an example of specifying a schedule using `cron` syntax. The following snippet specifies a watch to be triggered hourly at the 45th minute:

```
{
  "trigger" : {
    "schedule" : {
      "cron" : "0 45 * * * ?"
    }
  }
}
```

- `input` : This section is used to specify the input to load the data into the `Watcher` execution context. This data is referred to as **Watcher Payload** and will be available/accessible in subsequent phases of the

watcher execution so that it can be used to create conditions on it or used when generating actions. The payload can be accessed using the `ctx.payload.*` variable:

```
"input":{"search":{"request":{"indices":["logstash*"],"body":{"query":{"match":
{"message":"error"}}}}}}
```

As shown in the preceding code, an input of the `search` type is used to specify the query to be executed against Elasticsearch to load the data into Watcher Payload. The query fetches all the documents present in the indices of the `logstash*` pattern that contain `error` in the `message` field.

Note

Inputs of the `simple` type load static data, `http` loads an `http` response, and `chain` provides a series of inputs that can also be used in the `input` section.

- `condition`: This section is used to specify a condition against the payload in order to determine whether an action needs to be executed or not:

```
"condition":{"compare":{"ctx.payload.hits.total":{"gt":0}}}
```

As shown in the preceding code, it uses a condition of the `compare` type to determine whether the payload has any documents, and if it finds any, then the action will be invoked.

A condition of the `compare` type is used to specify simple comparisons, such as `eq`, `not-eq`, `gt`, `gte`, `lt`, and `lte`, against a value in the watch payload.

Note

Conditions of the `always` type always evaluate the watch condition to `true`, whereas `never` always evaluates watch condition to `false`. `array_compare` is used to compare against an array of values to determine the watches' condition, and `script` is used to determine the watches' condition.

- `actions`: This section is used to specify one or more actions that need to be taken when the watch condition evaluates to `true`:

```
"actions":{"log_error":{"logging":{"text":"The number of errors in logs is
{{ctx.payload.hits.total}}"}}}
```

As shown in the preceding code, it uses a logging action to log the specified text when the watch condition is met. The logs would be logged into Elasticsearch logs. The number of errors found is dynamically obtained using the field (`hits.total`) of the payload. The payload is accessed using the `ctx.payload.*` variable.

Note

Watcher supports the following types of actions: `email`, `webhook`, `index`, `logging`, `hipchat`, `Slack`, and `pagerduty`.

During the watches' execution, once the condition is met, a decision is made per configured action as to whether it should be throttled or continue executing the action. The main purpose of action throttling is to prevent too many executions of the same action for the same watch.

Watcher supports two types of throttling:

- **Time-based Throttling**: You can define a throttling period by using the `throttle_period` parameter as part of the action configuration or at the watch level (which applies to all actions) to limit how often the action is executed. The global default throttle period is `5` seconds.
- **ACK-based Throttling**: Using ACK Watch APIs, you can prevent watch actions from being executed again while the watch condition remains `true`.

Watches are stored in a special index named `.watches`. Every time a watch is executed, a `watch_record` containing details such as watch details, the time of watch execution, watch payload, and the result of the condition is stored in the watch history index, which is named `.watches-history-6-*`.

Note

A user with the `watcher_user` privilege can view watches and watch history.

Alerting in action

Now that we know what a **Watch** is made up of, in this section, we will explore how to create, delete, and manage watches.

You can create/delete/manage watches using the following software:

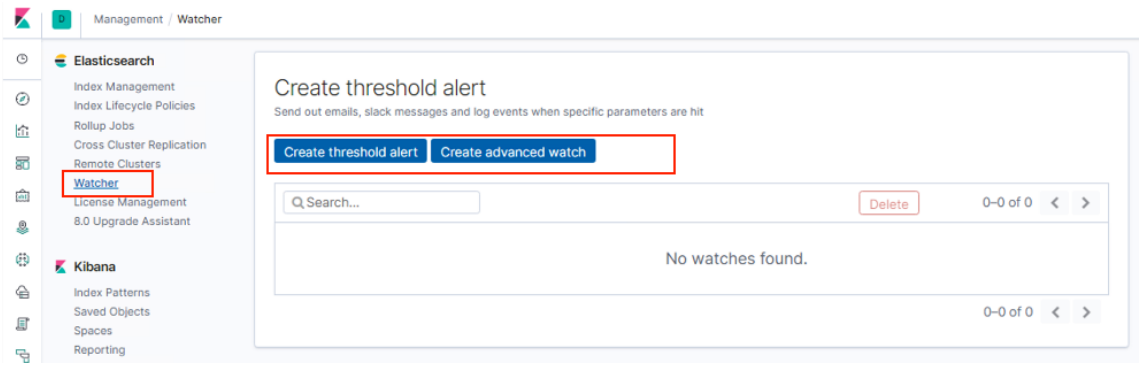
- Kibana Watcher UI
- X-Pack Watcher REST APIs

The **Watcher** UI internally makes use of Watcher REST APIs for the management of watches. In this section, we will explore the creation, deletion, and management of watches using the Kibana Watcher UI.

Creating a new alert

To create a watch, log in to Kibana (`http://localhost:5601`) as elastic/elastic and navigate to the `Management` UI; click on `Watcher` in the `Elasticsearch` section. Two options are available for creating alerts:

- `Create threshold alert`
- `Create advanced watch` :



By using the `Threshold alert` option, you can create a threshold-based alert to get notified when a metric goes above or below a given threshold. Using this UI, users can easily create threshold-based alerts without worrying about directly working with raw JSON requests. This UI provides options for creating alerts on time-based indices only (that is, the index has a timestamp).

Using the `Advanced watch` options, you can create watches by directly working with the raw `.json` required for the watches API.

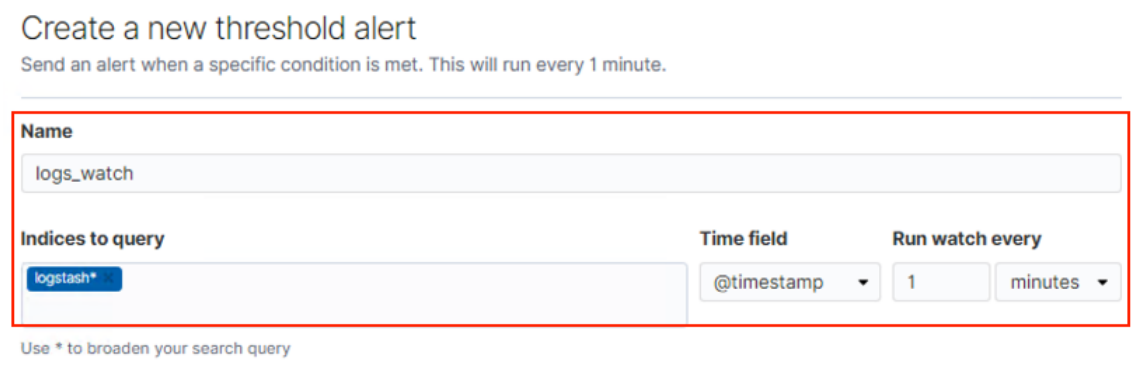
Note

The Watcher UI requires a user with `kibana_user` and `watcher_admin` privileges to create, edit, delete, and deactivate a watch.

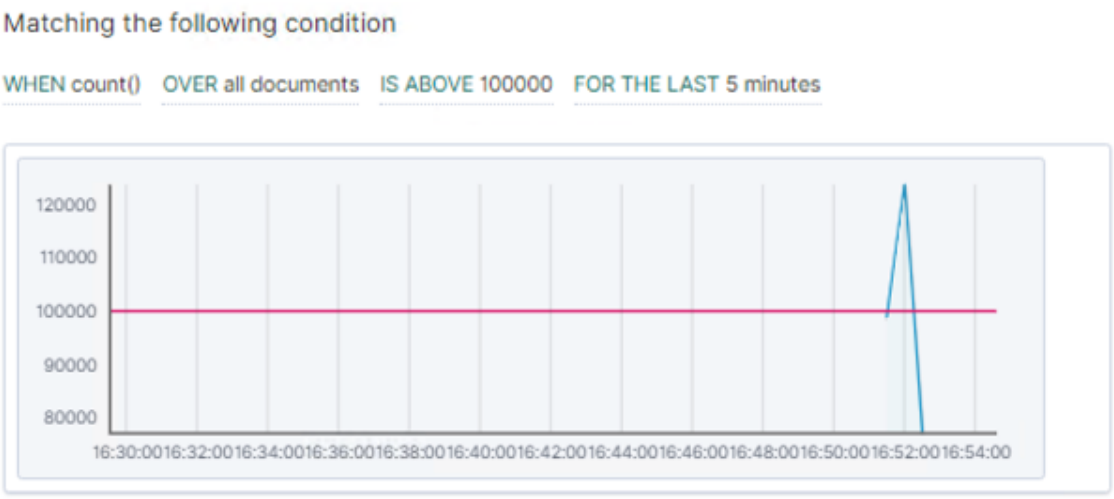
Threshold Alert

Click on `Create New Watch` and choose the `Threshold Alert` option. This brings up the `Threshold Alert` UI.

Specify the name of the alert; choose the index to be used to query against, the time field, and the trigger frequency in the `Threshold Alert` UI:



Then, specify the condition that will cause the alert to trigger. As the expressions/conditions are changed or modified, the visualization is updated automatically to show the threshold value and data as red and blue lines, respectively:



Finally, specify the action that needs to be triggered when the action is met by clicking on the `Add new action` button. It provides three types of actions, that is, email, slack, and logging actions. One or more actions can be configured:

Then, click on the `Save` button to create the watch.

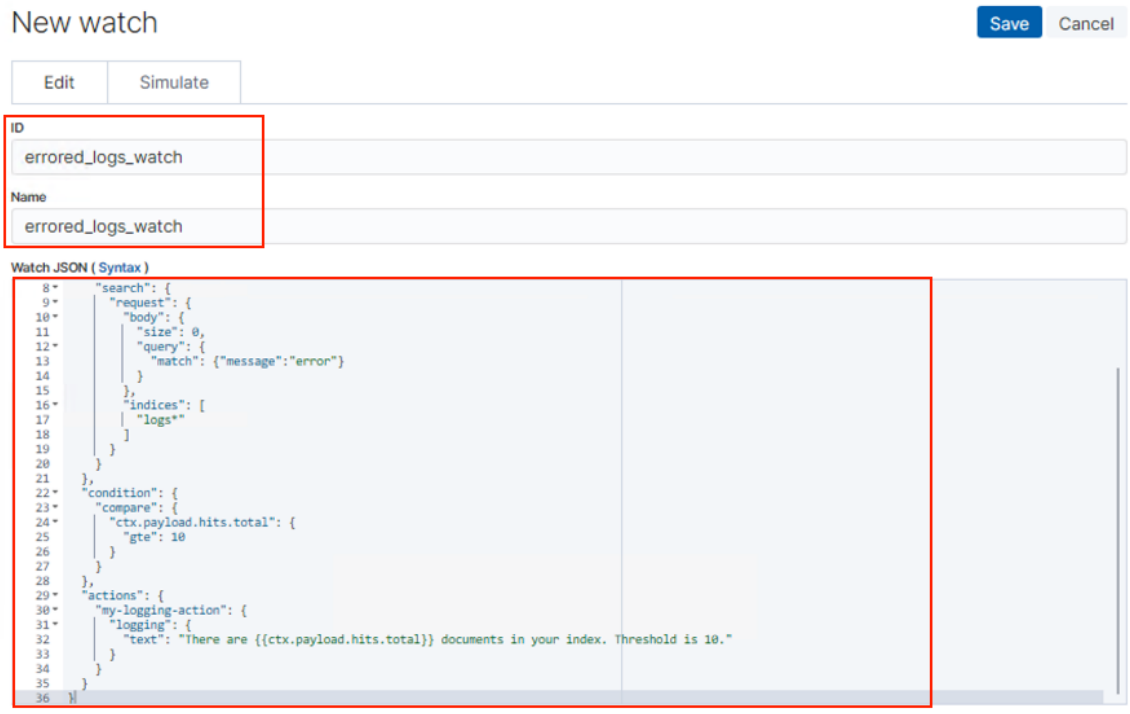
Clicking on `Save` will save the watch in the `watches` index and can be validated using the following query:

```
curl -u elastic:elastic -XGET http://localhost:9200/.watches/_search?
q=metadata.name:logs_watch
```

Advanced Watch

Click on the `Create New Watch` button and choose the `Advanced Watch` option. This brings up the `Advanced Watch` UI.

Specify the watch `ID` and watch `name`, and then paste the JSON to create a watch in the `Watch JSON` box; click on `Save` to create a watch. Watch ID refers to the identifier used by Elasticsearch when creating a Watch, whereas name is the more user-friendly way to identify the watch:



The `Simulate` tab provides a UI to override parts of the watch and then run a simulation of it.

Note

Watch Name will be stored in the metadata section of the watch body. You can use the metadata section when creating the watch to store custom metadata, tags, or information to represent/identify a watch.

Clicking on `Save` will save the watch in the `watches` index and can be validated using the following query:

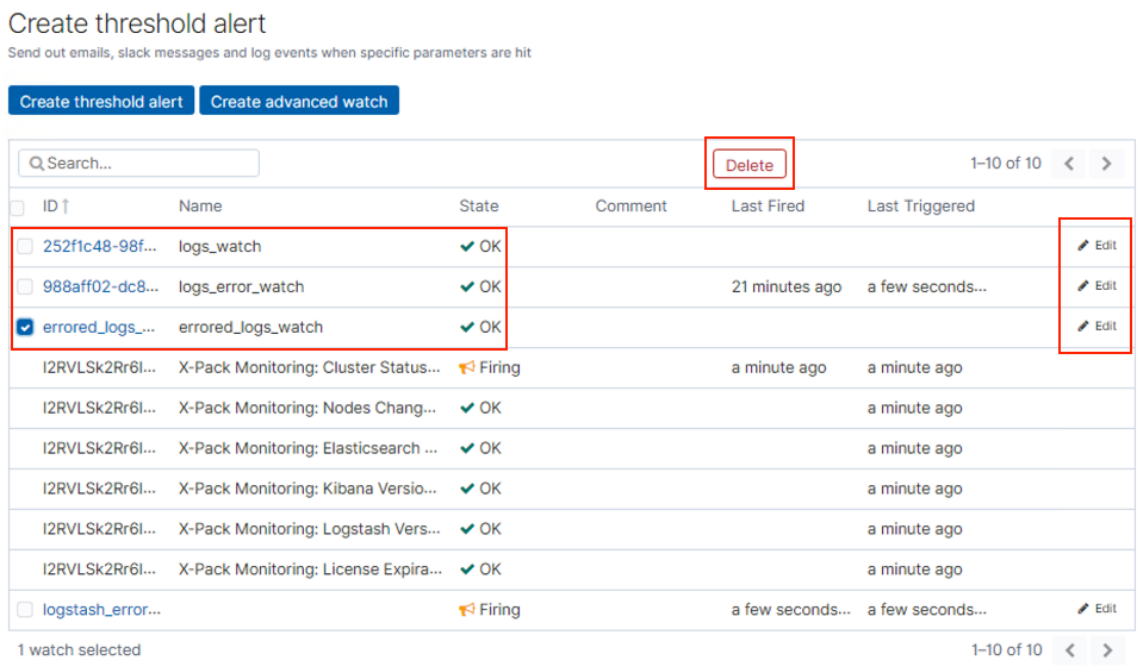
```
curl -u elastic:elastic -XGET http://localhost:9200/.watches/_search?
q=metadata.name:errored_logs_watch
```

Since we have configured logging as the action, when the alert is triggered, the same can be seen in `elasticsearch.log`:

```
[2019-05-23T17:43:07,809][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] The number of errors in logs is  39
[2019-05-23T17:43:37,379][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] The number of errors in logs is  39
[2019-05-23T17:44:07,343][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] The number of errors in logs is  39
[2019-05-23T17:44:37,380][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] The number of errors in logs is  39
[2019-05-23T17:45:07,373][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] The number of errors in logs is  39
[2019-05-23T17:45:37,393][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] The number of errors in logs is  39
[2019-05-23T17:46:04,394][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] There are 39 documents in your index. Threshold is 10.
[2019-05-23T17:46:07,383][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] The number of errors in logs is  39
[2019-05-23T17:46:37,428][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] The number of errors in logs is  39
```

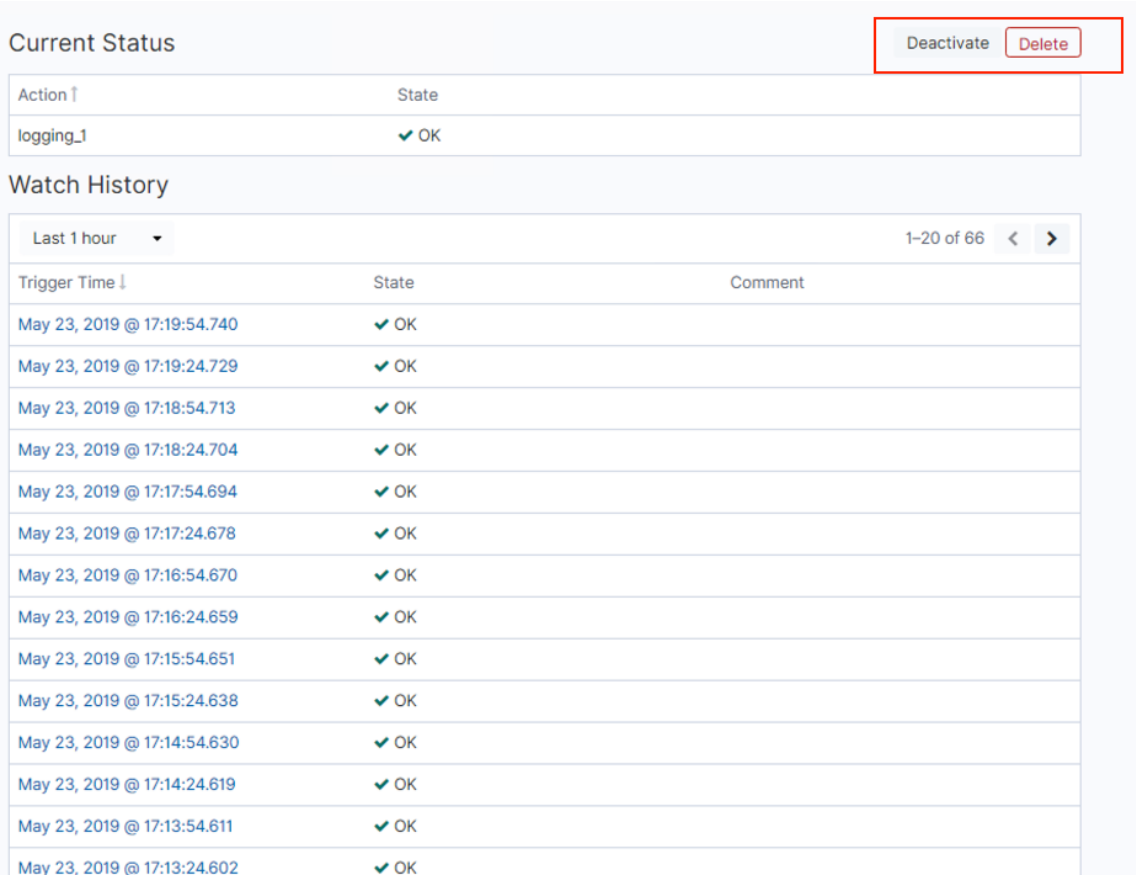
Deleting/deactivating/editing a watch

To delete a watch, navigate to the `Management` UI and click on `Watcher` in the `Elasticsearch` section. From the `Watches` list, select one or more watches that need to be deleted and click on the `Delete` button:



To edit a watch, click on the `Edit` link, modify the watch details, and click on the `Save` button to save your changes. To deactivate a watch (that is, to temporarily disable watch execution), navigate to the `Management` UI and click on `Watcher` in the `Elasticsearch` section. From the `Watches` list, click on the custom watch. The `Watch History` will be displayed. Click on the `Deactivate` button. You can also delete a watch from this screen.

Clicking on an execution time (link) in the `Watch History` displays the details of a particular `watch_record`:



Note

Starting from Elastic Stack 6.8.0 and 7.1.0, basic security features are free with the Elastic Basic License. More details can be found at https://www.elastic.co/blog/security-for-elasticsearch-is-now-free.

Summary

In this lab, we explored how to install and configure the X-Pack components in Elastic Stack and how to secure the Elastic cluster by creating users and roles. We also learned how to monitor the Elasticsearch server and alerting in order to generate notifications when there are changes or anomalies in the data.

In the next lab, we'll put together a complete application using Elastic Stack for sensor data analytics with the concepts we've learned about so far.