VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Akseli Ikkala

HÄIRIÖLOKIEN HALLINTA ELK-PINOLLA

Tekniikka
2021

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Akseli Ikkala
Opinnäytetyön nimi	Häiriölokien hallinta ELK-pinolla
Vuosi	2021
Kieli	suomi
Sivumäärä	53
Ohjaaja	Pirjo Prosi

Työn tarkoituksena oli tutkia, voidaanko ELK-pinoa hyväksikäyttäen kehittää Pesmel Oy käyttöön tuote jolla asiakkaille toimitetuista järjestelmistä voitaisiin kerätä häiriölokeja ja muuta niihin liittyvää tietoa yhteen keskitettyyn paikkaan.

Tällaisen tiedon kerääminen olisi hyvin hyödyllistä niin asiakkaiden kuin yrityksen oman henkilöstön puolesta. Asiakas pystyisi seuraamaan helposti toimitetun järjestelmän toimintaa ja kuormituskestävyyttä. Pesmelin oma henkilöstö taas pystyisi käyttämään kerättyä dataa mahdollisen kehitystyön ja vianetsinnän tukena.

Tutkimuksessa käytettiin ELK-pinoa keräämään ja visualisoimaan Pesmelin tarjoamaa WMS-testidataa tekstitiedostoista ja MSSQL-tietokannasta. Data kerättiin käyttäen Logstashia, jolle kehitettiin sopivat konfiguraatiot tähän käyttötarkoitukseen. Kerätty data lähetettiin Logstashilla Elasticsearchiin, josta sitä tutkittiin ja visualisoitiin käyttäen Kibanaa. Kibanassa testattiin myös usean käyttäjän järjestelmää, jossa käyttäjät pääsivät lukemaan vain tiettyä dataa.

Työn tuloksena luotiin toimiva testiympäristö, jolle syötettiin testidataa useamman WMS-järjestelmän simulaatioista. Testijärjestelmällä saatiin kerättyä ja visualisoitua tarjottua dataa onnistuneesti.

Avainsanat Logstash, Elasticsearch, Kibana, visualisointi, datan keräys

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Akseli Ikkala
Title	Error Log Control with ELK Stack
Year	2021
Language	Finnish
Pages	53
Name of Supervisor	Pirjo Prosi

This thesis work was done for Pesmel Oy in the need of a more flexible and easy-to- set- up and use platform for collecting, visualizing and exploring logs that Pesmel WMS system generates.

Collecting these logs would be beneficial for Pesmel and customers alike. Pesmel engineers could take advantage of ELK stacks capabilities to aggregate and visualize collected data in troubleshooting and customers could easily get information on the state of the system.

ELK Stack was used to collect and visualize test data from Pesmel WMS. Logstash was used to collect the data and custom Logstash configurations were developed for this particular use. The collected data was sent to from Logstash to Elasticsearch. From Elasticsearch the collected data was explored and visualized using Kibana. A multiple user Kibana system was also tested where some users had access only to certain data and visualizations in read only mode.

As a result of this project, a working test platform was built. Data from WMS simulations was collected by the test platform and visualizations were built on the collected data.

Keywords Logstash, Elasticsearch, Kibana, visualization, data collection

SISÄLLYS

KUVIOLUETTELO

1 JOHDANTO

Tiedon kerääminen, hakeminen ja visualisointi on nykypäivänä suuressa osassa yritysten tuotekehitystä ja asiakkaan kanssa kommunikointia. Hyvällä datan keräämisellä ja esittämisellä voidaan asiakkaalle esittää järjestelmän toimivuutta ja tarkastella mahdollisia toimintahäiriöitä.

Tämän työn tarkoituksena on tutkia, saataisiinko ELK-pinolla toteutettua tuote Pesmel Oy:n käyttöön, jolla voitaisiin kerätä asiakkaalle toimitetusta varastonhallintajärjestelmästä häiriölokeja ja muuta liittyvää tietoa yhteen keskitettyyn paikkaan.

Pesmel on vuonna 1978 perustettu vientiyritys, jonka pääasiallinen toimiala on metallituotteet. Yritys toimii myös paperi-, sellu- ja rengasaloilla. /1/

Yritys tuottaa automaattisia kuljetin-, varasto- ja pakkausjärjestelmiä teollisuuden tarpeisiin. Yrityksen viisi kulmakiveä ovat: innovaatio, insinööritaito, järjestelmä tietoisuus, räätälöinti ja elinikäinen tuki.

Yli neljänkymmenen vuoden kokemuksen aikana Pesmel on tuottanut yli 400 ratkaisua yrityksille viidellä maanosalla.

Vuonna 2019 Pesmellillä oli 144 työntekijää ja toimistoja Pohjois-Amerikassa, Aasiassa ja Euroopassa. Päämajaansa yhtiö pitää Suomessa Kauhajoella. /2/

ELK-pinolla toimiva kokonaisuus voitaisiin liittää erilliseksi osaksi varastonhallintajärjestelmää paikallisesti tai keskittää yksi alusta verkkopohjaiseksi pilvipalvelun, esimerkiksi AWS:n avulla, jolloin asiakkaalle toimitettuun järjestelmään tarvitsisi asentaa vain datankeräysosat ELK-pinosta. Haku ja visualisointi osat olisivat pilvessä, johon asiakas saisi omat tunnukset ja visualisoinnit.

Pesmel on kehittänyt erilaisia vastaavia räätälöitäviä tuotteita asiakkaille, mutta tarkoituksena olisi tutkia ja kehittää yhtenäinen tuote, joka voitaisiin minimaalisin

muutoksin ja kuormituksin liittää uuteen tai olemassa olevaan asiakkaalle toimitettuun järjestelmään.

Yhtenäinen ratkaisu olisi hyödyllinen niin Pesmelin tuotekehityksen kuin asiakkaidenkin kannalta, koska yhtenäisen tuotteen kehittäminen ja ylläpito on paljon yksinkertaisempaa kuin monen erillisen räätälöidyn ratkaisun. Tuotteen pilvimahdollisuudet ovat myös kiinnostavia mutta asettavat haasteita ja vaatimuksia tietoturvan osalta. Asiakkaan sisäisessä verkossa oleva ratkaisu olisi tietoturvallisempi mutta vaikeammin hallittava ja tiedon tarkastelu onnistuisi vain asiakkaan verkosta tai VPN:n ylitse etänä.

2 ELK-PINO

Elasticsearch B.V:n kehittämä ELK-pino on avoimen lähdekoodin alusta, johon kuuluvat kolme eri projektia: Logstash, Elasticsearch ja Kibana. Näillä eri komponenteilla voidaan kerätä, parsia, hakea ja visualisoida lähes mitä tahansa saatavilla olevaa tietoa. Alustan kaikki komponentit ovat saatavilla eri LINUX, MAC OS ja WINDOWS ympäristöihin. /3/ Elastic B.V tarjoaa myös tilausmallisia vaihtoehtoja, joilla ympäristöön saa enemmän ominaisuuksia.

Lisenssitasot ovat seuraavat: ”FREE AND OPEN”, ”BASIC”, ”GOLD”, ”PLATINUM” ja ”ENTERPRISE”. Ensimmäiset kaksi ovat ilmaisia ja avoimia. ”BASIC”-taso on lisensoitu Elastic-lisenssillä ja sisältää esimerkiksi normaalit tietoturvaominaisuudet, kun taas ”FREE AND OPEN”-taso on lisensoitu avoimella SSPL-lisenssillä. ”GOLD”-tasosta eteenpäin Elastic B.V vaatii käyttäjältä maksua. /26/

2.1 Logstash

Logstash on ELK-pinon osa, jolla dataa voidaan kerätä, parsia, koostaa ja siirtää Elasticsearchiin. Logstash rinnalle kuuluu myös Beats. Beats on pelkkä kevyt datankeräysosa, jolla voidaan syöttää dataa Logstashille tai suoraan Elasticsearchiin. Vaikka Beats voi syöttää dataa suoraan Elasticsearchiin yleensä tarvitaan kuitenkin Logstashin datan käsittelemiseen. Beatsin käyttö mahdollistaa datan keräyksen monelta laitteelta ilman, että järjestelmässä tarvitsisi olla montaa instanssia Logstashistä, joka syö resursseja paljon Beatsia enemmän.

Logstash ja Beats pystyvät lukemaan dataa monessa eri muodossa ja monesta eri järjestelmästä, tässä tutkielmassa tullaan kuitenkin keskittymään datan keräämiseen tekstitiedostoista ja MSSQL-tietokannasta. /4/

Kuten jo aiemmin mainittu, Beats on tarkoitettu vain datan keräykseen ja kuljettamiseen. Vain Logstash voi hoitaa datan muokkauksen, koostamisen ja parsimisen raakamuodosta indeksoitavaan ja haettavaan muotoon. Tämä on erittäin tärkeää, kun dataa kerätään tekstiedostoista, joissa data on yleensä vain merkkijonoina. MSSQL-tietokannan kohdalla keräys on suoraviivaisempaa, koska MSSQL on relaatiotietokanta ja data on jo valmiiksi indeksoitua. /5/

Logstash konfiguroidaan käyttäen konfigurointitiedostoja. Tiedostot on kirjoitettu Ruby-kielellä YAML-formaatissa. Konfiguraatiot muodostuvat yleensä kolmesta osasta: sisäänsyötöistä, suodattimista ja ulossyötöistä **(Kuvio 1.)**. /6/

```
# This is a comment. You should use comments to describe
# parts of your configuration.
input {
  ...
}

filter {
  ...
}

output {
  ...
}
```

Kuvio 1. Logstash-konfigurointitiedoston rakenne. /6/

Sisäänsyötöt ovat liitännäisiä, jotka lukevat dataa. Esimerkiksi SQL-liitännäinen. Liitännäisiä voi olla monta ja niille voidaan antaa omat ”nimet”, joilla ne on mahdollista erotella.

Suodattimet muuntavat ja parsivat dataa konfiguraation mukaan. Esimerkiksi merkkijonoista voidaan poimia halutut osat indekseihin, jotta ne ovat paremmin

haettavissa ja yhdistettävissä Elasticsearchissa. Parsimiseen voi käyttää esimerkiksi GROK-liitännäistä, joka etsii ehtojen mukaisesti haluttuja kenttiä tai merkkijonon muotoja ja poimii merkkijonosta tietyt palat indekseihin.

Ulossyötöt kertovat mihin data siirretään, kun se on ensin luettu ja suodatettu. Tähän käytetään yleensä Elasticsearchia. /7/

2.2 Elasticsearch

Elasticsearch on avoimen lähdekoodin analyyttinen moottori kaikentyyppiselle datalle. Logstash toimii ELK-pinossa tietokantana ja sinne varastoidaan esimerkiksi Logstashille kerätty ja muokattu data. Kun data on tallennettu Elasticsearchiin voidaan sitä vasten tehdä monimutkaisiakin kyselyjä ja tallennettuun tietoon perustuen voidaan luoda visualisointeja Kibanassa.

Elasticsearch tallentaa dataa JSON-dokumentteina ja näiden dokumenttien kokoelmaa kutsutaan indeksiksi. JSON-dokumentit ovat indekseissä ja Elasticsearch käyttää käänteistä indeksiä tiedon hakemiseen. Käänteiseen indeksiin on kirjattu kaikki sanat, jotka esiintyvät indekseissä.

Datan hakeminen indeksistä on hyvin nopeaa, lähes reaaliaikaista.

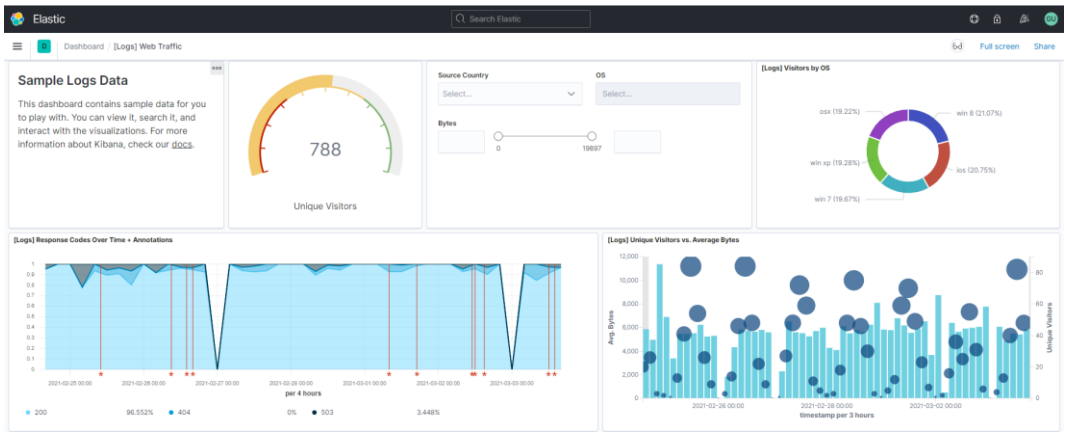
Elasticsearch tukee useita ohjelmointikieliä ja tekstikieliä, virallisesti tietokanta on käännetty 34-kielelle. /8/

Elasticsearch konfiguroidaan toimintaan konfigurointitiedostoilla, jotka on kirjoitettu YAML-formaatissa. Elasticsearch toimitetaan jo valmiiksi toimivalla konfiguraatiolla mutta yleensä konfiguraatiota joudutaan räätälöimään edes jonkin verran tiettyyn käyttötarkoitukseen. /9/

2.3 Kibana

Kibana on avoimen lähdekoodin visualisointiohjelmisto, joka on tiiviisti integroitu Elasticsearchiin. Kibana tarjoaa haku ja visualisointitoimintoja Elasticsearchiin tallennettuun tietoon.

Kibanaa käytetään tiedon katselemiseen, etsimiseen ja visualisointiin. Kibanaan voidaan luoda näkymiä, joita kutsutaan Kibanassa nimellä ”dashboard”. Näkymiin voidaan asettaa erilaisia visualisointeja kuten diagrammeja tai taulukoita **(Kuvio 2.)**. /10/



Kuvio 2. Kibanan esimerkkinäkymä.

Kibana tarjoaa myös mahdollisuuden suorittaa kyselyjä Elasticsearchista. Näissä käytetään Kibana Query Language (KQL)-syntaksia. Mikäli käyttäjä ei siis löydä haluamaansa tietoa valmiista näkymästä, voi hän yrittää tehdä erilaisia hakuja. KQL-kyselyjä voidaan myös tallentaa myöhempää katselua varten. /11/

Kibanan kautta voidaan myös hallita Elasticsearchin käyttäjiä ja indeksejä, Elasticsearch tarjoaa API:n myös tekstipohjaiseen käyttöön tätä varten. /10/

3 TYÖN ALOITTAMINEN

Vaatimukset käytettävälle alustalle ovat:

- Datan keräys lokitiedostoista.
- Datan keräys MSSQL-tietokannasta.
- Datan esittäminen visuaalisesti kaavioiden avulla.
- Tiedon hakeminen kerätystä datasta.

Logstash pystyy keräämään dataa tekstitiedostoista ja SQL-tietokannoista, MSSQ-kyselyt tapahtuvat JDBC-ajurin avulla, jota ei toimiteta Logstashin mukana, mutta on saatavilla vapaasti internetistä. /12/

Kibanan avulla on mahdollista lukea tietoa, jonka Logstash on kerännyt ja lähettänyt Elasticsearchiin. Kuten jo aiemmin luvussa 2.3 mainittu, Kibanaan pystyy luomaan omia ”näkymiä” ja Elasticsearchiin talletettua dataa voidaan vapaasti kysellä Kibanan kautta.

ELK-pino täyttää aloitusvaatimukset ja sen pohjalle voidaan luoda testiympäristö.

Työ aloitetaan tutustumalla ELK-pinoon kuuluvien komponenttien asennukseen ja käyttöön. Aivan ensimmäisenä on haettava Elastic B.V:n verkkosivulta haluttu versio työhön tarvittavista komponenteista, jonka jälkeen eri komponentit Logstash, Elasticsearch ja Kibana täytyy konfiguroida vuorollaan haluttuun käyttötarkoitukseen. Näistä suuritöisimmät tulevat olemaan Logstash ja Kibana, koska näihin täytyy luoda asioita käsin alusta asti. Elasticsearchin konfigurointi on enimmäkseen asetusten muuttamista.

3.1 ELK-pinon asennus

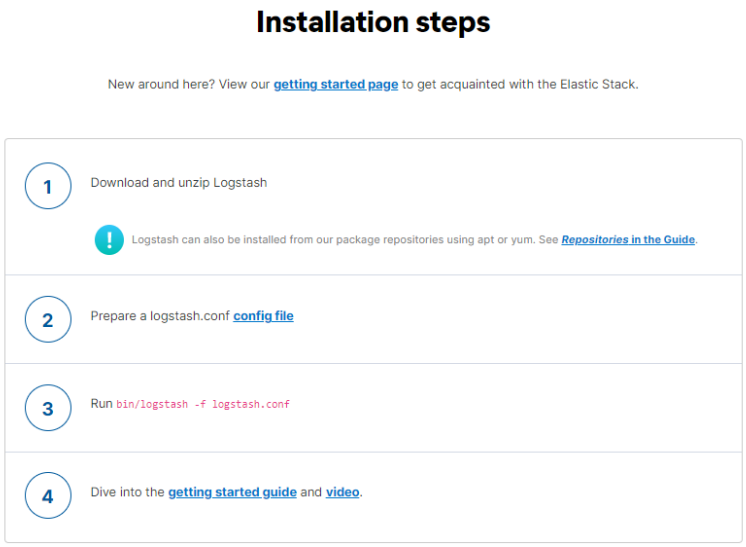
Työssä käytettävät ohjelmat löytyvät internetistä kehittäjän eli Elastic B.V:n sivulta ja ovat ladattavissa ZIP-arkistoina. Työ tullaan tekemään Windows-ympäristössä,

jolloin ZIP-arkisto on ainut jakelumuoto. Poikkeuksena Elasticsearch, joka voidaan asentaa myös MSI-asennuspakettina. Käytämme kuitenkin työssä ZIP-arkistoja. LINUX-ympäristössä on mahdollista ladata ohjelmat pakettihallintaohjelmien kautta. /13/

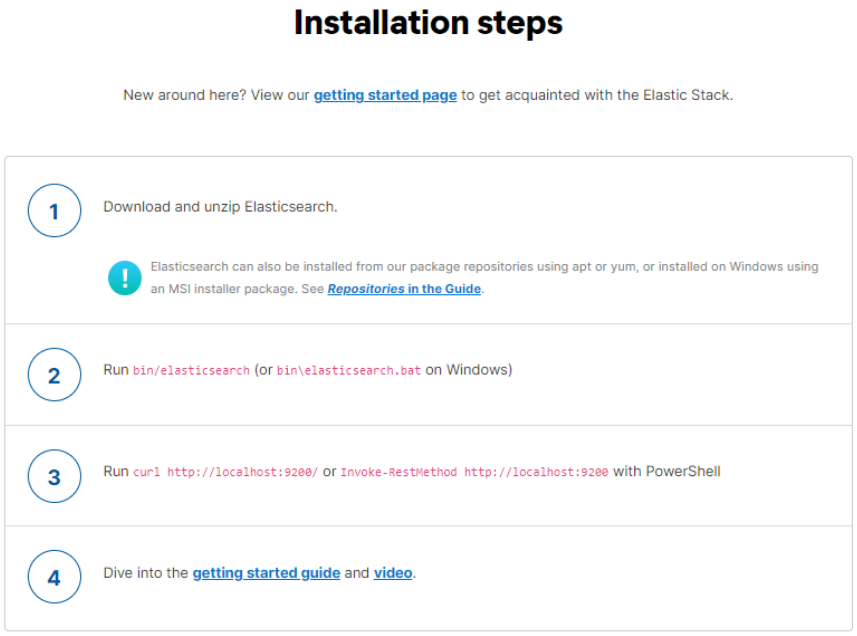
ELK-pinon voi myös tilata pilvessä olevana toteutuksena suoraan Elasticilta. Tässä vaihtoehdossa ohjelmia ei tarvitse ladata paikallisesti tietokoneelle ja Elastic myös hoitaa alustan pystytyksen. Konfigurointi on tietenkin edelleen käyttäjän vastuulla. /14/

Työn tutkimus- ja testiluonteen vuoksi ei alustaa lähdetä pystyttämään pilveen, vaan luodaan paikallinen ympäristö, jolla voidaan tutkia konseptin toimivuutta paremmin pienessä mittakaavassa.

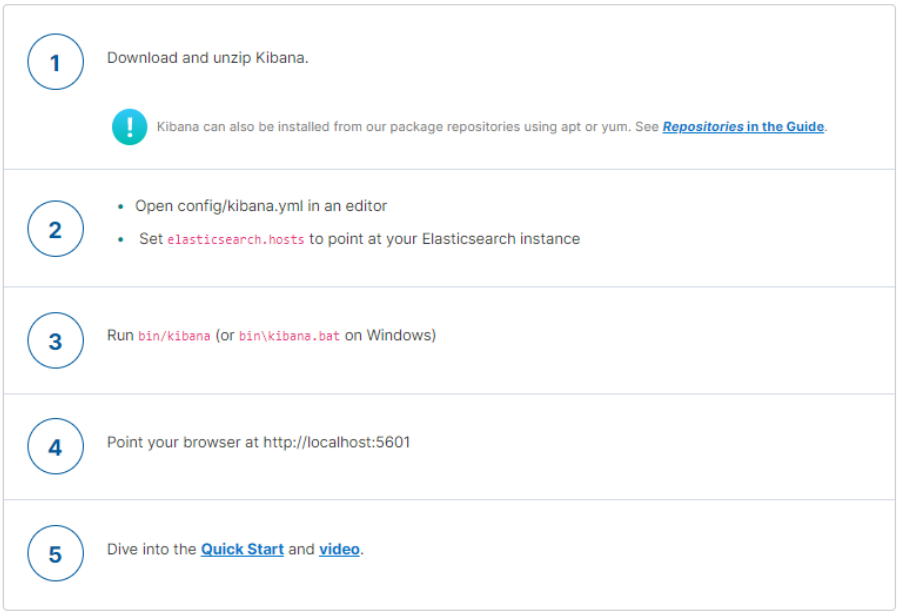
Kun kaikkien ohjelmien ZIP-arkistot on ladattu, puretaan ne haluttuun polkuun Windowsissa ja tämän jälkeen voidaan siirtyä konfigurointiin. Varsinaista asentamista ei ole, vaan ohjelmat toimivat purkamisen jälkeen, kunhan ne on ensin konfiguroitu oikein **(Kuviot 3–5.)**.



Kuvio 3. Logstashin asennusvaiheet. /15/



Kuvio 4. Elasticsearchin asennusvaiheet. /13/



Kuvio 5. Kibanan asennusvaiheet. /16/

3.2 Ohjelmien testaus

Ohjelmat on hyvä testata niiden latauksen ja purkamisen jälkeen ja tarkastaa, että ne toimivat oikein eikä lataamisessa syntynyt tiedostovirheitä.

3.2.1 Logstashin testaus

Luodaan Logstashille testikonfiguraatio ja käynnistetään Logstash. Testikonfiguraatio vain tulostaa tiedostosta tekstiä komentoriville eikä vielä lähetä sitä mihinkään. Ei myöskään konfiguroida vielä minkäänlaisia suodattimia kerätylle datalle, vaan otetaan koko rivi kerrallaan ja tulostetaan se komentoriville **(Kuvio 6.)**.

```
input {
  file {
    path => "C:/Users/Ikkal/Desktop/log.txt"
    start_position => "beginning"
  }
}
filter {

}
output {
 stdout{
  codec => rubydebug
  }
}
```

Kuvio 6. Logstashin testikonfiguraatio.

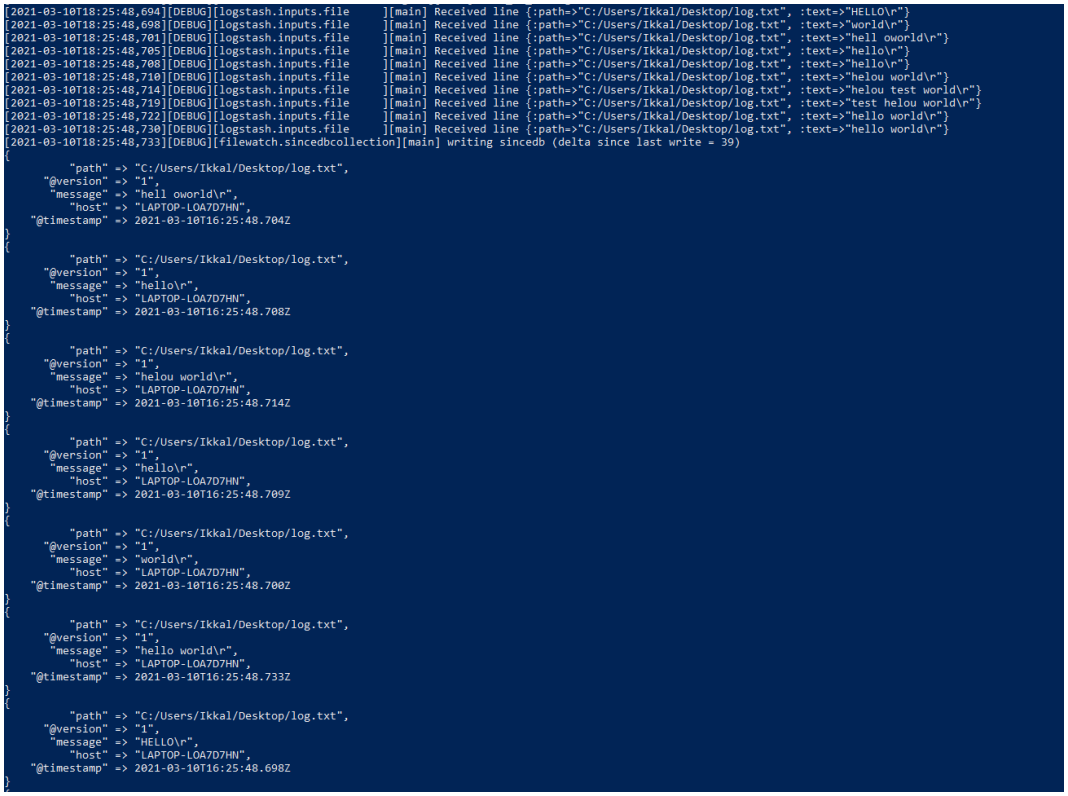
Kaikkia ELK-komponentteja kannattaa Windows-ympäristössä suorittaa PowerShellin kautta. Avataan seuraavaksi PowerShell ja siirrytään oikeaan hakemistoon.

Logstash käynnistetään hakemistosta ”Logstash/bin” komennolla ”Logstash”.

Lisätään komentoon lippu -f ja sen perään haluttu konfiguraatiotiedosto, tässä tapauksessa test_conf.conf. Tämä lippu kertoo Logstashille, että halutaan käyttää tiettyä konfiguraatiotiedostoa **(Kuvio 7.)**.

```
PS C:\Users\Ikkal\Desktop\ELK\logstash-7.4.0\bin> .\logstash -f .\test_conf.conf
```

Kuvio 7. Logstashin käynnistyskomento.

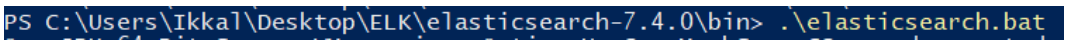


Kuvio 8. Logstashin vianmääritysrivit ja tulostus.

Mikäli testi onnistui, Logstash ilmoittaa ”[DEBUG]”-merkatuilla riveillä vastaanotetut rivit ja tulostaa ne sen jälkeen komentoriville. Logstash tulostaa myös META-tiedot, jotka liitetään jokaiseen riviin ohjelman toimesta. Vain ”message”-rivi oli kirjoitettuna testi tiedostoon **(Kuvio 8.)**.

3.2.2 Elasticsearchin testaus

Elasticsearchia ei tarvitse konfiguroida ennen käynnistystä, ainakaan tätä testiä varten. Elasticsearch on helpoin testata avaamalla PowerShell-ikkuna ”Elasticsearch/bin”-hakemistossa ja suorittamalla ”elasticsearch.bat”-tiedoston **(Kuvio 9.)**.



Kuvio 9. Elasticsearchin käynnistyskomento.

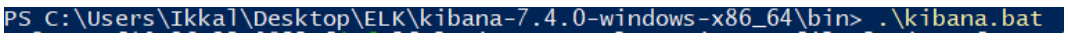
Mikäli Elasticsearch ei kirjoita virheitä ja viesti ”started” ilmestyy PowerShell-ikkunaan, voidaan todeta Elasticsearchin käynnistyneen onnistuneesti **(Kuvio 10.)**.

Kuvio 10. Elasticsearchin käynnistysviesti.

Elasticsearch on nyt käynnissä ja sinne voidaan lähettää dataa tai lukea sitä.

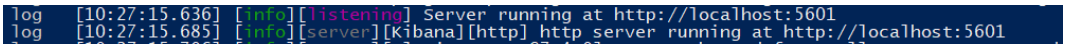
3.2.3 Kibanan testaus

Kibanaa ei tarvitse ainakaan testausta varten konfiguroida mitenkään ennen käynnistämistä. Kibana käynnistetään avaamalla PowerShell ja siirtymällä polkuun ”kibana/bin” ja suorittamalla ”kibana.bat”-tiedoston **(Kuvio 11.)**.



Kuvio 11. Kibanan käynnistyskomento.

Kuten edellisten ohjelmien kanssa alkaa Kibana tulostaa PowerShell-ikkunaan vianselvitys rivejä ja näistä ilmenee myös, milloin ja jos Kibana on onnistuneesti käynnistynyt **(Kuvio 12.)**.

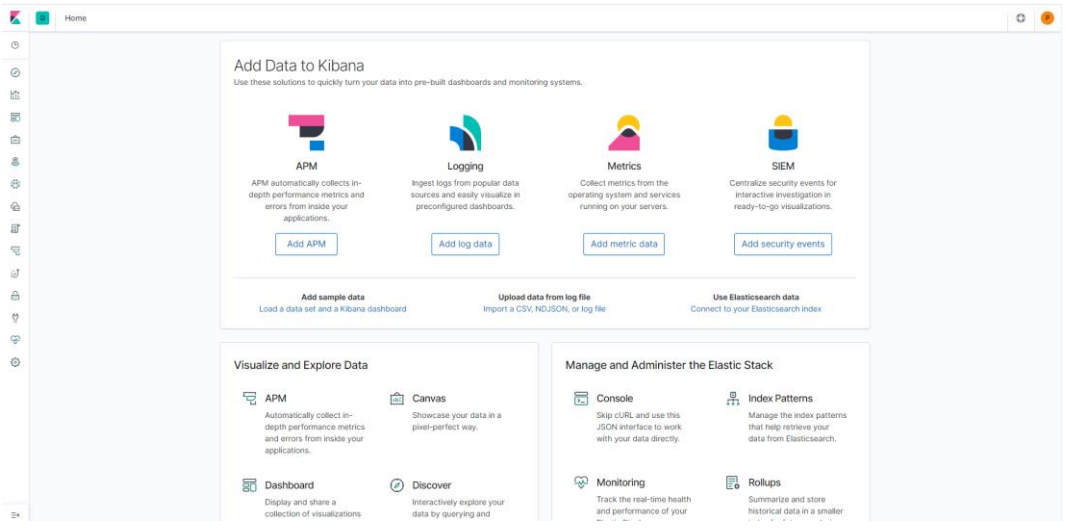


Kuvio 12. Kibanan käynnistysviesti.

Kun Kibana ilmoittaa PowerShell-ikkunassa, että serveri on käynnissä määrätyssä osoitteessa ja portissa on käynnistys onnistunut.

Tässä tapauksessa Kibanaan saa yhteyden osoitteesta ”http://localhost:5601”. Tämä tarkoittaa siis sitä, että Kibana-serveri on sidottu paikallisen koneen osoitteeseen eli ”localhost” ja porttiin 5601.

Kun tähän osoitteeseen siirtyy internet selaimella, pääsee Kibanan käyttöliittymään.



Kuvio 13. Kibanan oletuskotisivu.

Kibanan käyttöliittymä on onnistuneesti käynnistetty ja testi on onnistunut. Oletuksena Kibanan aloitussivulla on muutamia mahdollisuuksia ladata testidataa ja työpöytiä uusille kehittäjille, joiden avulla järjestelmään pääsee tutustumaan **(Kuvio 13.)**.

4 LOGSTASH - DATAN KERÄYS, SUODATUS JA LÄHETTÄMINEN

Pesmel varastonhallinta kirjoittaa lokirivejä MSSQL-tietokantaan ja useampaan tekstitiedostoon. Näistä lähteistä data täytyy kerätä, suodattaa ja siirtää Elasticsearchiin. Tähän käytetään Logstashia.

4.1 Datan keräys Logstashilla

Kerättävä data on tekstitiedostoina ja MSSQL-tietokannassa. Tekstitiedostoja on useampi, mutta datan muoto on sama kaikissa tiedostoissa. MSSQL-tietokannasta luetaan kahta eri taulua.

Näille eri lähteille täytyy luoda oma ”putki” eli ”pipeline”. Käytännössä tämä tarkoittaa sitä, että luodaan monta erillistä konfigurointitiedostoa Logstashille ja sen jälkeen ne lisätään ”Pipelines”-tiedostoon, joka annetaan Logstashille käynnistysargumentiksi. /17/ Kaikki nämä erilliset ”putket” käynnistetään siis yhdellä Logstashin instanssilla ja kerätty data ohjataan jokaisessa omaan Elasticsearchin indeksiin.

4.1.1 Datan keräys MSSQL-serveriltä

Data kerätään MSSQL-serveriltä SQL-kyselyinä. Jotta Logstash voi suorittaa SQL-kyselyitä tarvitaan JDBC-ajuri, joka toimii rajapintana Logstashin ja MSSQL-serverin välillä.

Sopiva JDBC-ajuri on saatavilla vapaasti Microsoftin verkkosivuilta. /18/ Ajuri ladataan ZIP-arkistona ja asetetaan käyttäjän valitsemaan paikkaan paikallisessa järjestelmässä, jossa myös Logstash sijaitsee. Logstashin konfiguraatiotiedostossa asetetaan JDBC-ajurin polku, JDBC-ajurin luokka ja tietokantayhteyden asetukset.

Vähintään on tarpeellista asettaa IP-osoite, josta tietokanta löytyy. Lisäksi tarvitaan kirjautumistiedot tietokantaan ja suoritettava SQL-lause.

Logstash-konfiguraatiotiedostoon voidaan asettaa JDBC-lisäosalle muitakin asetuksia. Esimerkiksi ajastus kuinka usein kysely suoritetaan tai sarake, josta seurataan mikä rivi tietokannasta on jo luettu. /19/

Tämän työn jokaiseen Logstashin JDBC-liitännäisen konfiguraatioon on asetettu seuraavat asetukset:

- **jdbc_driver_library**: Polku, jossa JDBC-ajuri sijaitsee **(Kuvio 14.)**.
- **jdbc_driver_class**: Mitä ajuria kirjastosta käytetään. JDBC-ajurin ZIP-paketissa on Java-kirjasto, joka sisältää monta eri ajuria erilaisilla tietokannoille **(Kuvio 14.)**.
- **jdbc_connection_string**: Tietokantapalvelimen osoite ja tietokannan nimi
- **jdbc_user**: Tietokannan käyttäjätunnus **(Kuvio 14.)**.
- **jdbc_password**: Tietokannan käyttäjätunnuksen salasana **(Kuvio 14.)**.
- **Schedule**: Määrittelee, kuinka usein SQL-kysely suoritetaan. Käyttää CRON-ajastusta ja ”* * * * *” tarkoittaa ”joka minuutilla” **(Kuvio 14.)**.
- **statement**: Suoritettava SQL-lause. Lauseissa oleva ”:sql_last_value” muuttuja saadaan edellisen suorituksen metadatasta, tämä määritellään seuraavalla kolmella asetuksella **(Kuvio 14.)**.
- **use_column_value**: Logstash-parametri, jolla liputetaan, että rivien seuraamiseen halutaan käyttää arvoa tietystä sarakkeesta **(Kuvio 14.)**.
- **tracking_column**: Sarake, josta Logstashin halutaan kirjaavan ylös mitkä rivit on jo luettu, käytännössä tallennetaan aina suorituksen lopussa metadataksi viimeisin ID-sarakkeen arvo **(Kuvio 14.)**.
- **last_run_metadata_path**: Tiedosto, josta Logstash löytää edellisen suorituksen tiedot **(Kuvio 14.)**.

```
input {
 jdbc {
  jdbc_driver_library => "C:\Users\wms\Desktop\ELK\JDBC\sqljdbc_4.2\enu\jre8\sqljdbc42.jar"
  jdbc_driver_class => "com.microsoft.sqlserver.jdbc.SQLServerDriver"
  jdbc_connection_string => "jdbc:sqlserver://DESKTOP-UIDV0HM:1433;databaseName=WMS_database_7404;"
    jdbc_user => "user"
    jdbc_password => "password"
  schedule => "* * * * *"
    statement => "select * from log where id > :sql_last_value"
    use_column_value => true
    tracking_column => id
    last_run_metadata_path => "C:\Users\wms\.logstash_jdbc_last_run3"
    }
}
```

Kuvio 14. Logstashin JDBC-liitännäisen konfigurointi.

4.1.2 Datan keräys lokitiedostoista

Datan keräys lokitiedostoista on helpompaa kuin MSSQL-tietokannasta, mutta asia käy suodattamisessa paljon monimutkaisemmaksi johtuen lokitiedostojen rakenteesta, joissa lokirivit ovat vain yksittäisiä merkkijonoja tietoa tai monen rivin pituisia ”stack dump” lokituksia, joita tulee monesti ohjelmien kohdatessa virhetilanteita ja silloin lokiin tulostetaan koko ohjelman pino, joka on yleensä monen rivin mittainen.

Rivien kerääminen lokitiedostoista on Logstashilla helppoa. Logstash sisältää valmiiksi liitännäisen ”file”, jolla tekstitiedostoja voidaan lukea.

”File”-liitännäiselle täytyy vähintään antaa ”path”-parametri, joka kertoo missä luettava tiedosto sijaitsee. Kun ”path”-parametri on asetettu, luetaan halutusta tekstitiedostosta dataa rivi kerrallaan. Työssä luettavissa lokeissa kuitenkin voi olla usean rivin tapahtumia, joten tarvitaan lisää parametrejä, joilla määritellään näiden tapausten oikea käsitteleminen. /20/

```
input {
  file {
    path => "C:/temp/wms_rolling_log.log"
    sincedb_path => "C:\Users\wms\.sincedb1"
    start_position => "beginning"
    codec => multiline {
          pattern => "^\["
          negate => true
          what => "previous"

  }
  tags => "wms"
}
}
```

Kuvio 15. Logstashin lokitiedostojen lukukonfiguraatio.

Tämän työn lokitiedostojen lukukonfiguraatiossa on käytetty samoja parametrejä, jotka ovat seuraavat:

- **path**: Polku mistä luettava tiedosto löytyy **(Kuvio 15.)**. /20/
- **sincedb_path**: Kertoo minne viimeisen suorituksen metadata on tallennettu, Logstash käyttää metadatan tallentamiseen ”sincedb”-nimisiä tiedostoja, joihin on kirjattu ylös esimerkiksi mikä on viimeisin luettu rivi tiedostossa, jokaiselle ”input”-konfiguraatiolle täytyy olla määriteltynä oma ”sincedb”-polku **(Kuvio 15.)**. /20/
- **start_position**: Määrittelee ensimmäisessä käynnistyksessä, mistä kohdasta tiedostoa rivejä aletaan lukea. Yleensä määritellään ”beginning” eli tiedoston alusta. Ensimmäisen suorituksen jälkeen tätä ei oteta huomioon **(Kuvio 15.)**.
- **codec**: Määritellään syötettävien rivien koodekki. ”multiline” tarkoittaa, että yksi merkintä voi olla monen rivin kokoinen **(Kuvio 15.)**. /21/

- **pattern**: Sisältyy ”codec”-valintaan, tähän kirjoitetaan säännöllisenä lausekkeena eli ”regex” (Regular Expression language)-koodina malli. Mallin avulla tunnistetaan, kuuluuko kyseinen rivi useamman rivin viestiin vai onko se yhden rivin viesti. Tässä tapauksessa etsitään, onko rivin alussa merkki ” [”, koska kaikki yhden rivin viestit alkavat kyseisellä merkillä **(Kuvio 15.)**. /21/
- **negate**: Sisältyy ”codec”-valintaan, tällä parametrilla määritellään mitä ”pattern”-tunnistus tekee. Kun ”negate” on määritelty todeksi eli ”true”, kaikki rivit, joita ei tunnisteta annetulla mallilla kuuluvat silloin monen rivin käsittävään viestiin **(Kuvio 15.)**. /21/
- **what**: Sisältyy ”codec”-valintaan, määrittelee mitä tehdään, jos rivi tunnistetaan kuuluvaksi usean rivin viestiin. ”previous”-valinta liittää rivin edelliseen luettuun riviin, tätä tehdään niin monta riviä, kunnes tunnistus ei enää päde **(Kuvio 15.)**. /21/
- **tags**: Määrittää luetuille riveille metadataksi halutun ”tagin”, joilla rivit voidaan tunnistaa. Suurin hyöty saadaan, kun samassa konfigurointitiedostossa on monta ”input”-lisäosaa käytössä **(Kuvio 15.)**. /20/

4.2 Datan suodatus Logstashilla

Yleensä kerätylle datalle täytyy tehdä ”suodattamista”, ennen kuin se lisätään Elasticsearchiin. Suodatus tehdään Logstash-konfiguraatiotiedoston ”filter” -osiossa ja siinä käytetään suodatusliitännäisiä.

Suodattamisessa kerätyn datan kenttiä muunnetaan erilaiseen muotoon tai kerätystä datasta voidaan poistaa tietoa, joka ei ole olennaista. Dataa voidaan myös jäsentää tai siihen voidaan lisätä metatietoa, jonka käyttäjä on määritellyt. /22/

Suodattimia voi olla Esimerkiksi GROK-suodatin, jolla voidaan kerätä jäsentämättömistä merkkijonoista tiettyjä osia ja muuttaa data järjestetyksi ja haettavaksi. /23/

4.2.1 Lokitiedostojen datan suodatus

Lokitiedostoista kerätylle datalle täytyy tehdä suodatusta käyttäen GROK-suodatinta ja date-suodatinta. MSSQL-serverillä olevaa dataa ei tarvitse ajaa GROK-suodattimen läpi, koska data on relaatiotietokannassa valmiiksi jäsennetyssä muodossa.

GROK-suodatin toimii säännönmukaisilla lausekkeilla eli regex-koodilla.

Säännönmukaisella lausekkeella kerrotaan GROK-suodattimelle runko, jota data noudattaa ja miten data täytyy jäsentää. Data voidaan leikata paloiksi esimerkiksi tiettyjen merkkien kohdalta tai välistä **(Kuvio 16.).** /23/

Kaikki lokitiedostoihin kirjoitettava data vastaa tiettyä tyyliä, suodattimen ”match”-kohdalle annetaan rungoksi muoto:

”{"mes-
sage"=>"\[(?<LOG>[^\]]+)\]\[(?<LEVEL>[^\]]+)\]%{TIMESTAMP_ISO8601:TIMESTAMP} \[(?<PROCESS>[^\]]+)\] %{WORD:POINT} %{GREEDYDATA:MESSAGE}" }”.

Kyseinen regex-koodi avattuna tarkoittaa seuraavaa:

- **\ [(?<LOG>[^\]]+)\]**: Lokitiedoston nimi, alkaa merkillä ”[” ja loppuu merkkiin ”]”.
- **\[(?<LEVEL>[^\]]+)\]**: Lokirivin taso esimerkiksi ”DEBUG” tai ”ERROR”. Alkaa merkillä ”[” ja loppuu merkkiin ”]”.
- **%{TIMESTAMP_ISO8601:TIMESTAMP}**: Lokirivin aikaleima.
- **\[(?<PROCESS>[^\]]+)\]**: Prosessi, josta lokirivi on kirjoitettu. Alkaa merkillä ”[” ja loppuu merkkiin ”]”.
- **%{WORD:POINT} %**: Paikka prosessissa. Yksittäinen sana.
- **%{GREEDYDATA:MESSAGE}"**: Itse rivin sisältämä viesti, voi olla ilmoitus tai virheteksti. Tähän otetaan kaikki lopputeksti, jota rivissä on jäljellä. **(Kuvio 16.)**

Filter-osiossa on käytetty myös ”date”-suodatinta, joka täsmäyttää GROK-suodattimen ”TIMESTAMP”-kohdan aikaleiman Logstashin metatietojen ”date”-kenttään. Tämän avulla voidaan datan hakemiseen ja järjestämiseen Elasticsearchista käyttää päivänmääriä. Mikäli tätä suodatusta ei tehdä, saisi data aikaleimakseen Logstashin keräysajankohdan eikä riviin merkatun aikaleiman ajankohtaa **(Kuvio 16.)**.

```
filter {
    grok {
      match => { "message" => "\[(?<LOG>[^\]]+)\]\[(?<LEVEL>[^\]]+)\] %{TIMESTAMP_ISO8601:TIMESTAMP} \[(?<PROCESS>[^\]]+)\] %{WORD:POINT} %{GREEDYDATA:MESSAGE}" }
    }

    date{
      match => [ "TIMESTAMP", "yyyy-MM-dd HH:mm:ss.SSS" ]

  }
}
```

Kuvio 16. Lokitiedostoa lukevan konfiguraation filter-osio.

4.2.2 Tietokannasta kerätyn tiedon suodatus

MSSQL-tietokannasta kerätylle datalle ei ole tarpeellista tehdä samanlaista GROK-suodatusta kuin lokitiedostoista kerätylle. MSSQL on relaatiotietokanta ja data,

jota siitä kerätään, on valmiiksi jäsennettynä. Tietokannasta kerätylle datalle täytyy ensin tehdä ”mutate”-suodatus, jolla tietokannasta tulevan datan aikaleima muutetaan merkkijonoksi ja sen jälkeen tehdään ”date”-suodattimella operaatio, jolla aikaleima muutetaan samaan muotoon kuin lokitiedostoissa oleva. Tämän jälkeen ”date”-suodatin täsmäyttää ”time_stamp”-kentän tiedon, joka tulee kerätystä datasta Logstashin metatietojen ”@timestamp”-kenttään ja poistaa kerätyistä tiedosta ”time_stamp”-kentän **(Kuvio 17.)**.

Näin saadaan aikaleima vastaamaan samaa muotoa kuin lokitiedostoissa ja aikaleiman sisältämä ajankohta saadaan Logstashin metatietoihin, joita käytetään datan hakemiseen ja järjestämiseen myöhemmin. Mikäli tätä suodatusta ei tehtäisi olisi aikaleima väärässä kentässä ja aikaleiman ajankohta olisi Logstashin suoritusajankohta eikä aikaleimaan merkattu ajankohta.

```
filter{
 mutate{
    convert => ["time_stamp", "string"]
  }
 date{
    match => ["time_stamp" , "ISO8601", "yyyy-MM-dd HH:mm:ss.SSS"]
    target => "@timestamp"
    remove_field => [ "time_stamp" ]
  }
}
```

Kuvio 17. Tietokantaa lukevan konfiguraation filter-osio.

4.3 Datan lähetys Logstashilla

Datan lähettäminen tapahtuu Logstashin konfigurointitiedoston ”output” -osiossa. Data lähetetään tässä työssä Elasticsearchiin, mutta Logstash pystyy lähettämään keräämäänsä dataa myös muualle. Data voidaan lähettää esimerkiksi tekstitiedostoon, tietokantaan tai sähköpostiin. /25/ Varsinkin tekstitiedostoon kirjoittaminen on hyödyllistä, kun uusia Logstash-konfiguraatioita kehitetään. Tällöin mahdollisesti virheellistä dataa ei tarvitse lähettää Elasticsearchiin.

Työssä data lähetetään Elasticsearchiin, tähän käytetään Elasticsearch-liitännäistä. Liitännäiselle kerrotaan missä osoitteessa Elasticsearch sijaitsee, mihin indeksiin data halutaan kirjoittaa, mikä tunnustieto datalle annetaan Elasticsearchissa ja Elasticsearchin käyttäjä sekä salasana **(Kuvio 18.)**.

Mikäli määriteltyä indeksiä ei ole valmiiksi olemassa Elasticsearchissa, kyseisen niminen indeksi luodaan automaattisesti. Käyttäjä määritellään yleensä erilliseksi ”kirjoittaja”-käyttäjäksi, joka pystyy vain kirjoittamaan dataa Elasticsearchiin, mutta ei pääse tekemään muita operaatioita.

Tunnustieto vastaa SQL-serveristä pääavainta, joka on aina uniikki ja jolla rivit voidaan yksilöidä toisistaan. Tunnustieto voidaan automaattisesti määritellä Logstashin luomasta META-tiedosta datalle.

Salasana voidaan antaa selvämerkkisenä konfigurointitiedostossa tai sille voidaan luoda ”holvi” Elasticsearchiin, josta salasana haetaan. Tämä ei ole pakollista varsinkaan, jos ELK-pino pyörii suljetussa tuotantoverkossa, mutta se parantaa tietoturvaa.

Jotta avainholvia ja muita tietoturvaominaisuuksia voidaan hyödyntää, täytyy lisenssitaso nostaa oletuksena olevasta ”FREE”-tasosta ”BASIC”-tasoon.

```
output{
 elasticsearch {
    hosts => ["localhost:9200"]
    index => "cs_log_alarm_sql"
    document_id => "%{id}"
    user => logstash_internal
    password => "${es_pass}"
  }
}
```

Kuvio 18. Logstashin ”output”-osio.

4.4 Logstash putkilinjat ja usean konfiguraatiotiedoston käyttö

Monesti on käytännöllistä käyttää useampaa konfigurointitiedostoa, jos dataa pitää kerätä monesta paikasta ja erilaisilla ehdoilla. Kaiken keräämisen pystyy yhdistämään yhteen tiedostoon, mutta silloin täytyy käyttää esimerkiksi merkkauksia ”input”-liitännäisten kohdalla ja vertailla niitä eri kohdissa myöhemmin konfigurointitiedostossa. Tätä metodia käytettäessä voivat konfigurointitiedostot paisua tarpeettoman suuriksi ja vaikeasti luettaviksi **(Kuvio 19.)**.

Tämän takia helpointa on luoda useampi konfigurointitiedosto ja käyttää Logstashin putkilinjaominaisuutta, jolla useampia konfigurointitiedostoja voidaan käyttää yhtä aikaa samassa Logstashin instanssissa.

```
input {
  file {
    path => "C:/temp/wms_rolling_log.log"
    sincedb_path => "C:\Users\Simulation\Desktop\ELK\.sincedb1"
    start_position => "beginning"
    codec => multiline {
          pattern => "^\["
          negate => true
          what => "previous"

  }
  tags => "wms"
}

  file {
    path => "C:/temp/uiapi_rolling_log.log"
    sincedb_path => "C:\Users\Simulation\Desktop\ELK\.sincedb2"
    start_position => "beginning"
    codec => multiline {
          pattern => "^\["
          negate => true
          what => "previous"

  }
  tags => "uiapi"
}
}

filter {
    grok {
      match => { "message" => "\[(?<LOG>[^\]]+)\]\[(?<LEVEL>[^\]]+)\] %{TIMESTAMP_ISO8601:TIMESTAMP} \[(?<PROCESS>[^\]]+)\] %{WORD:POINT} %{GREEDYDATA:MESSAGE}" }
    }

    date{
      match => [ "TIMESTAMP", "yyyy-MM-dd HH:mm:ss.SSS" ]

  }
}

output {
if "wms" in [tags] {
elasticsearch {
    hosts => ["localhost:9200"]
    index => "cs_log_wms"
        user => logstash_internal
    password => "${es_pass}"
  }
}

if "uiapi" in [tags] {
elasticsearch {
    hosts => ["localhost:9200"]
    index => "cs_log_uiapi"
        user => logstash_internal

    password => "${es_pass}"
  }
}
}
```

Kuvio 19. Yksi kokonainen Logstash-konfigurointitiedosto, jossa useampi ”input”-liitännäinen ja useampi ”output”-liitännäinen.

Putkilinjat eli ”pipelines” toimivat niin, että ”pipelines.yml”-tiedostoon lisätään putkilinja. Putkilinjalle kirjoitetaan nimi, mahdolliset asetukset ja polku, josta konfigurointitiedosto löytyy **(Kuvio 21.)**. Käynnistyksessä Logstashille ei tarvitse antaa parametreja vaan ”pipelines.yml”-tiedosto luetaan automaattisesti, kun Logstash käynnistetään ilman parametrejä

Käynnistys tapahtuu samaan tapaa kuin Kuviossa 7, mutta parametri osuus ”-f .test_con.conf” jätetään pois. Käynnistyksen jälkeen pitäisi ikkunaan tulostua viesti, jossa Logstash ilmoittaa lukevansa ”pipelines.yml”-tiedostoa **(Kuvio 20.)**.

```
[DEBUG][logstash.config.source.multilocal] Reading pipeline configurations from YAML
```

Kuvio 20. Logstash-putkilinjojen lukeminen.

Kuviossa 20 on ensimmäiseen putkilinjaan annettu asetuksena ”pipeline.workers” ja sen arvoksi 2. Tämä asetus määrittää, kuinka monta suorittimen säiettä Logstash käyttää. Tätä asetusta muuttamalla voidaan säädellä Logstashin kuormitusta ja tehokkuutta järjestelmässä. /27/ Lopuille putkilinjoille on annettu asetus ”queue.type” ja sen arvoksi ”persisted” jolla määritellään, että Logstash kirjoittaa lukupuskurin levylle. Kun arvona on ”persisted” ja Logstash sammuisi äkillisesti, voidaan lukupuskuri palauttaa ja toimintaa jatkaa vähäisillä riskeillä datan menettämisestä.

```
- pipeline.id: runningLogs
  pipeline.workers: 2
  path.config: "/Users/Simulation/Desktop/ELK/logstash-7.4.0/config/tata-wms-log.conf"

- pipeline.id: SQLalarm
  queue.type: persisted
  path.config: "/Users/Simulation/Desktop/ELK/logstash-7.4.0/config/sql-alarm.conf"

- pipeline.id: SQLlog
  queue.type: persisted
  path.config: "/Users/Simulation/Desktop/ELK/logstash-7.4.0/config/sql-log.conf"

- pipeline.id: SQLtask
  queue.type: persisted
  path.config: "/Users/Simulation/Desktop/ELK/logstash-7.4.0/config/sql-task.conf"

- pipeline.id: SQLloadmove
  queue.type: persisted
  path.config: "/Users/Simulation/Desktop/ELK/logstash-7.4.0/config/sql-loadmove.conf"
```

Kuvio 21. Logstash-putkilinjoja.

5 ELASTICSEARCH JA KIBANA KONFIGUROINTI

5.1 Elasticsearchin konfigurointi

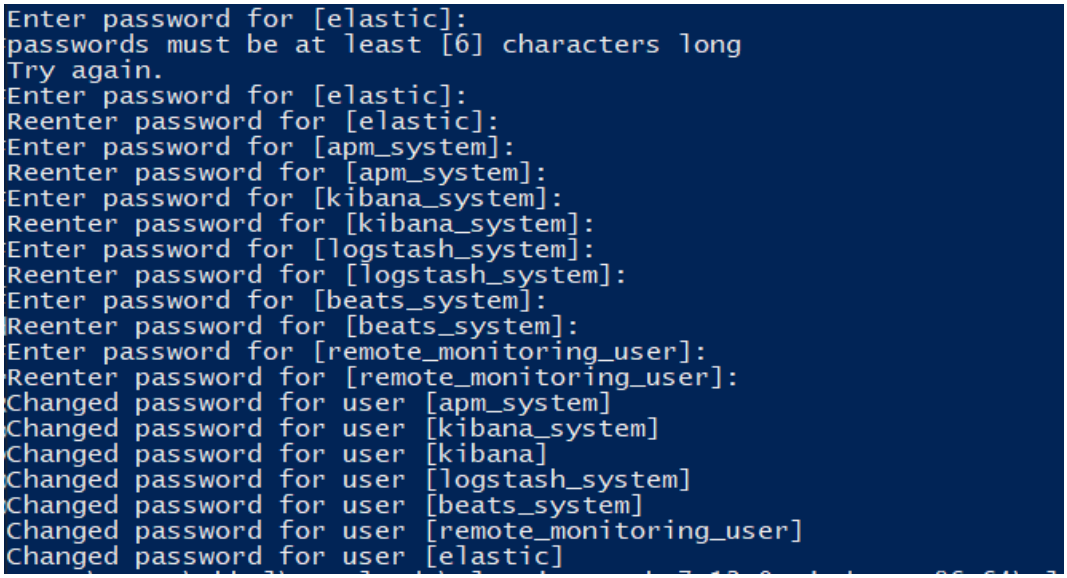
Elasticsearchia ei tarvitse paljoa konfigurointia.

Laitetaan päälle normaalit turvaominaisuudet, jotka saa käyttöön asettamalla Elasticsearch konfiguraatiotiedostoon ”elasticsearch.yml”-rivin.

”xpack.security.enabled: true”-rivillä saadaan koko ELK-pinoon normaalit turvaominaisuudet käyttöön. Samalla myös lisenssi päivittyy ”FREE”-tasosta ”BASIC”-tasoon.

Luodaan myös salasanat Elasticsearchin sisäänrakennetuille käyttäjille. Tämä tapahtuu ajamalla komennon ”elasticsearch-setup-passwords” ”Elasticsearch/bin”-kansiossa. ”elasticsearch-setup-passwords” komentoon pitää vielä lisätä valinta ”auto” tai ”interactive”. ”auto”-valinta luo salasanat automaattisesti satunnaisista merkeistä ja ”interactive”-valinnalla käyttäjä saa määritellä salasanat itse **(Kuvio 22.)**. /28/

Luodut salasanat laitetaan muistiin, ”elastic”-käyttäjän salasana on varsinkin tärkeä. Kyseisiä tunnuksia tarvitaan Kibanan konfiguroinnissa.

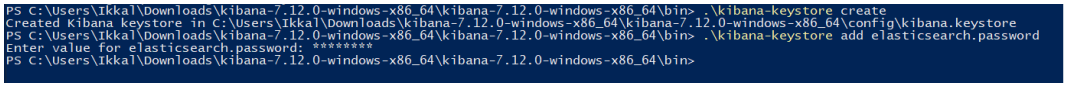


Kuvio 22. Elasticsearchin sisäänrakennettujen käyttäjien salasanojen asettaminen.

5.2 Kibanan konfigurointi

Kibanan käytön aloittamiseen ei ole tarvetta suuriin muutoksiin konfiguroinnissa, ainoa asia mitä Kibanan konfigurointiin on tarvittavaa tehdä, on lisätä luvussa 5.1 konfiguroitu ”elastic”-käyttäjän salasana Kibanan avainholviin ja lisätä ”elastic”-käyttäjä Kibanan konfigurointitiedostoon.

Avataan PowerShell-ikkuna tiedostopolussa ”/kibana/bin” ja kirjoitetaan komento ”kibana-keystore create” ja sen jälkeen lisätään salasana avainholviin komennolla ”kibana-keystore add elasticsearch.password”. Syötetään salasana ohjelman sitä kysyessä ja salasana on nyt lisätty Kibanan avainholviin **(Kuvio 23.)**. /28/



Kuvio 23. Kibanan avainholvin luominen ja avaimen lisääminen.

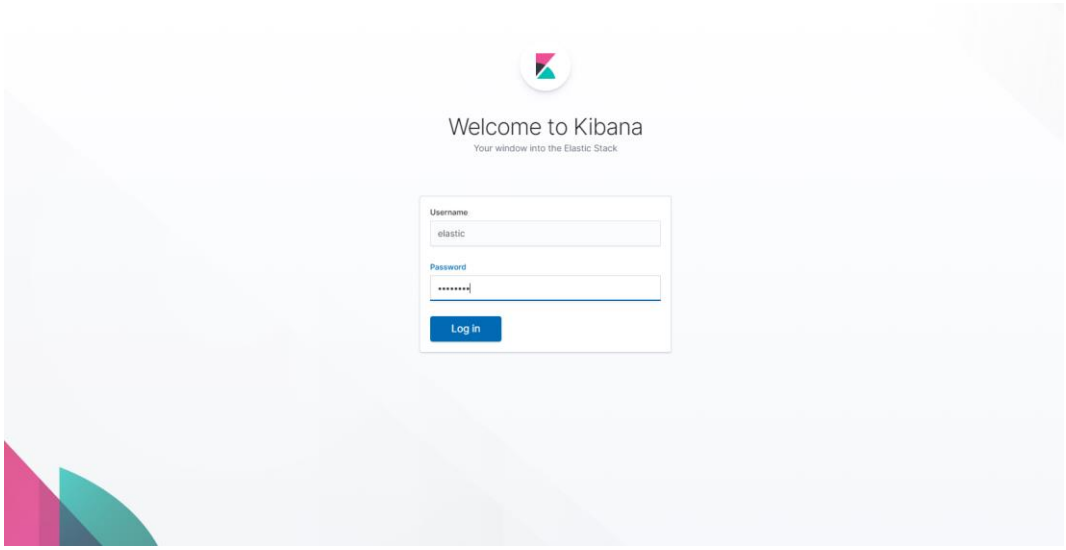
Tämän jälkeen avataan tiedostopolusta ”kibana/config” tiedosto ”kibana.yml”, tässä tiedostossa on Kibanan määritykset. Tiedostoon on valmiiksi määritelty Kibanalle portti kohdassa ”server.port” ja osoite kohdassa ”server.host”, mutta ne on kommentoitu pois etumerkillä ”#”. Oletuksena portti on ”5601” ja osoite on ”localhost”. Otetaan näistä kohdista kommenttimerkki ”#” pois, jolloin rivit saadaan käyttöön. Kohdassa ”server.name” otetaan kommenttimerkki pois ja lisätään arvoksi ”WMS-TEST”, tämä on Kibana-serverin nimi. Viimeiseksi kohdassa ”elasticsearch.username” otetaan kommenttimerkki pois ja muutetaan arvoksi ”elastic” **(Kuvio 24.)**. /28/

```
# Kibana is served by a back end server. This setting specifies the port to use.
server.port: 5601

# Specifies the address to which the Kibana server will bind. IP addresses and host names are both valid values.
# The default is 'localhost', which usually means remote machines will not be able to connect.
# To allow connections from remote users, set this parameter to a non-loopback address.
server.host: "localhost"

# Enables you to specify a path to mount Kibana at if you are running behind a proxy.
# Use the `server.rewriteBasePath` setting to tell Kibana if it should remove the basePath
# from requests it receives, and to prevent a deprecation warning at startup.
# This setting cannot end in a slash.
#server.basePath: ""

# Specifies whether Kibana should rewrite requests that are prefixed with
# `server.basePath` or require that they are rewritten by your reverse proxy.
# This setting was effectively always `false` before Kibana 6.3 and will
# default to `true` starting in Kibana 7.0.
#server.rewriteBasePath: false

# The maximum payload size in bytes for incoming server requests.
#server.maxPayloadBytes: 1048576

# The Kibana server's name.  This is used for display purposes.
server.name: "WMS-TEST"

# The URLs of the Elasticsearch instances to use for all your queries.
#elasticsearch.hosts: ["http://localhost:9200"]

# When this setting's value is true Kibana uses the hostname specified in the server.host
# setting. When the value of this setting is false, Kibana uses the hostname of the host
# that connects to this Kibana instance.
#elasticsearch.preserveHost: true

# Kibana uses an index in Elasticsearch to store saved searches, visualizations and
# dashboards. Kibana creates a new index if the index doesn't already exist.
#kibana.index: ".kibana"

# The default application to load.
#kibana.defaultAppId: "home"

# If your Elasticsearch is protected with basic authentication, these settings provide
# the username and password that the Kibana server uses to perform maintenance on the Kibana
# index at startup. Your Kibana users still need to authenticate with Elasticsearch, which
# is proxied through the Kibana server.
elasticsearch.username: "elastic"
```

Kuvio 24. Osa Kibanan konfigurointiedostosta muutoksineen.

Kibana on alustavasti konfiguroitu ja sen käyttö voidaan aloittaa. Kun Kibana käynnistetään ensimmäisen kerran ja turvallisuusominaisuudet on aktivoitu, täytyy sisään kirjautua ”elastic”-käyttäjällä. ”elastic”-käyttäjälle asetettiin salasana luvussa

5.1. /28/ Kibanaan pääsee nyt siirtymällä internet selaimella osoitteeseen ”localhost:5601” **(Kuvio 25.)**.



Kuvio 25. Kibanan kirjautumisikkuna.

6 KIBANA – HALLINTA JA VISUALISOINTI

Kuten mainittu luvussa 2.3 Kibana on osa ELK-pinoa, jolla voidaan hakea, visualisoida ja hallita tietoa Elasticsearchissa. Kibanaa käytetään myös käyttäjätunnusten, roolien ja lupien hallintaan. Kibanan sisälle voidaan luoda erilaisia näkymiä, hakuja ja tiloja, joilla voidaan hallita mitä dataa on missä ja kenen näkyvillä.

6.1 Kibanan tilat

Luodaan kaksi erillistä uutta tilaa oletustilan lisäksi. Erillisillä tiloilla voidaan kontrolloida tietoa johon käyttäjät pääsevät käsiksi, tilojen luonnissa ja myöhemmin asetusten kautta on mahdollista määrittää mitä ominaisuuksia tiloissa on käytettävissä. Esimerkiksi tietystä tilasta voidaan rajoittaa kokonaan visualisoinnit pois ja antaa pelkät asetukset ja klusterin hallintaan vaikuttavat työkalut, kun taas toiseen tilaan voidaan määrittää pelkät visualisoinnit näkyviin. Jokaisessa tilassa voi myös olla määriteltynä omat visualisoinnit ja kustomoinnit. Molemmille ”asiakkaiden”-käyttäjille annetaan omat tilansa ja näin he näkevät vain itselleen tarkoitetut tiedot ja visualisoinnit.

Asetuksia, tiloja, käyttäjiä ja rooleja yms. pääsee hallitsemaan joko Kibanan oman ohjelmointirajapinnan kautta tai Kibanan käyttöliittymästä. Tässä työssä on katettu käyttöliittymän kautta hallitseminen.

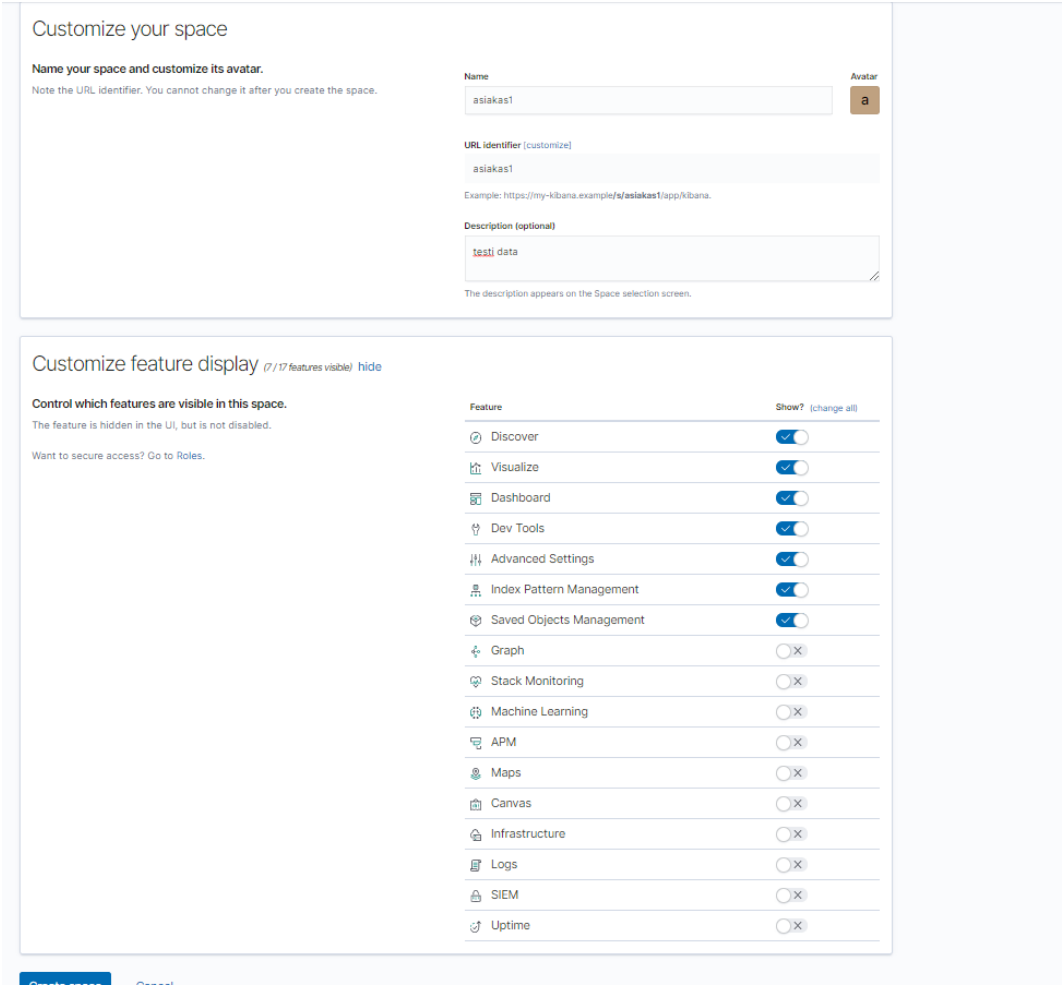
Kibanan käyttöliittymässä kirjautumisen jälkeen aukeaa käyttäjälle määritetty aloitusnäkymä ja tila, tässä ”elastic”-käyttäjän tapauksessa tila on Kibanan oletustila ja sivu on kotisivu.

Tilan oikeassa reunassa on kuvakkeita, joista voi siirtyä eri sivuille tilan sisällä. Alimmaisimpana oletustilassa on asetukset.

Asetuksissa kohdan ”Kibana” alla on kohta ”Spaces”, tämän valitsemalla päästään hallitsemaan tiloja ja luomaan uusia. Valitsemalla ”Create new” päästään luomaan

uusi tila, luodaan kaksi erinimistä tilaa samoilla ominaisuuksilla. Tilaa luodessa annetaan ensin kohtaan ”Name” nimi, ”URL identifier”-kenttä täyttyy tämän perusteella automaattisesti.

”Customize feature display”-kohdassa lukee oletuksena ”17/17 feature visible”, avataan valikko kohdasta ”show” ja poistetaan valinta kaikista muista kohdista paitsi ”Discover”, ”Visualize”, ”Dashboard”, ”Advanced Settings”, ”Index Pattern Management” ja ”Saved Objects Management” **(Kuvio 26.)**. Mikäli jokin ominaisuus ei tilan asetuksissa ole määritelty näkyväksi, on se vielä käytössä, jos käyttäjällä on siihen oikeudet. Lopuksi valitaan ”Create space”.

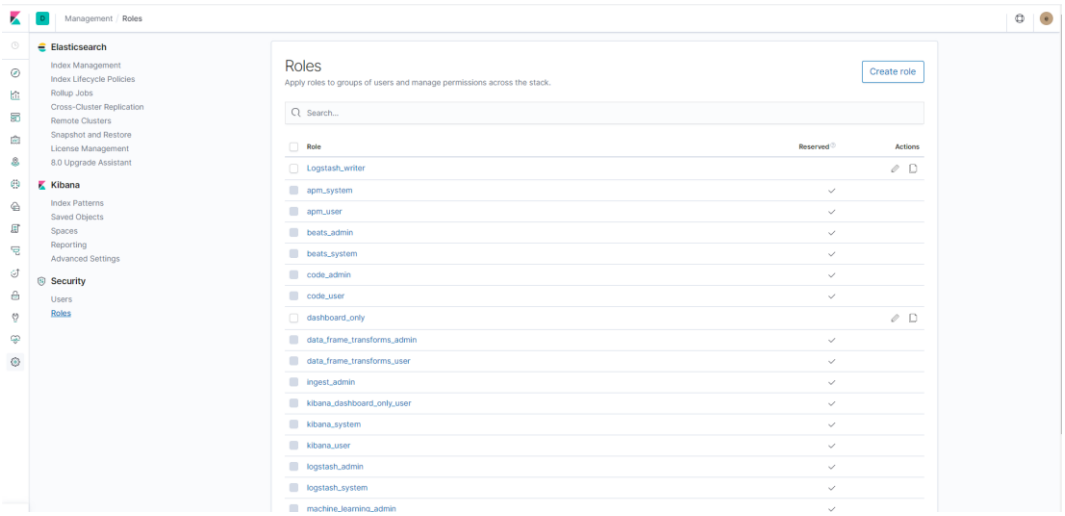


Kuvio 26. Uuden tilan luominen.

6.2 Käyttäjät ja roolit

Yleisesti on järkevää luoda eritasoisia käyttäjiä ja erilliset käyttäjät ja roolit sisäänrakennetun ”elastic”-käyttäjän rinnalle. Näin voidaan rajoittaa tietoa johon käyttäjät pääsevät käsiksi eikä esimerkiksi Logstash-käyttäjällä ole mitään syytä päästä lukemaan indeksien tietoja.

Logstashia varten luodaan oma rooli ja käyttäjä, samoin asiakkaita varten. Asiakkaiden käyttäjistä tulee vain lukutilassa olevat käyttäjät. Ensimmäisenä luodaan roolit. Asetus sivulla ”Security”-kohdassa alhaisimpana löytyvät roolit eli ”roles”. Tämän valitsemalla päästään hallitsemaan olemassa olevia rooleja ja luomaan uusia. Kibanassa tulee valmiiksi jo joitain määritettyjä rooleja, jotka ovat varattuja ja niitä ei voi poistaa tai muokata. Varatut roolit liittyvät ELK-pinoon valmiiksi sisäänrakennettuihin ominaisuuksiin **(Kuvio 27.)**.

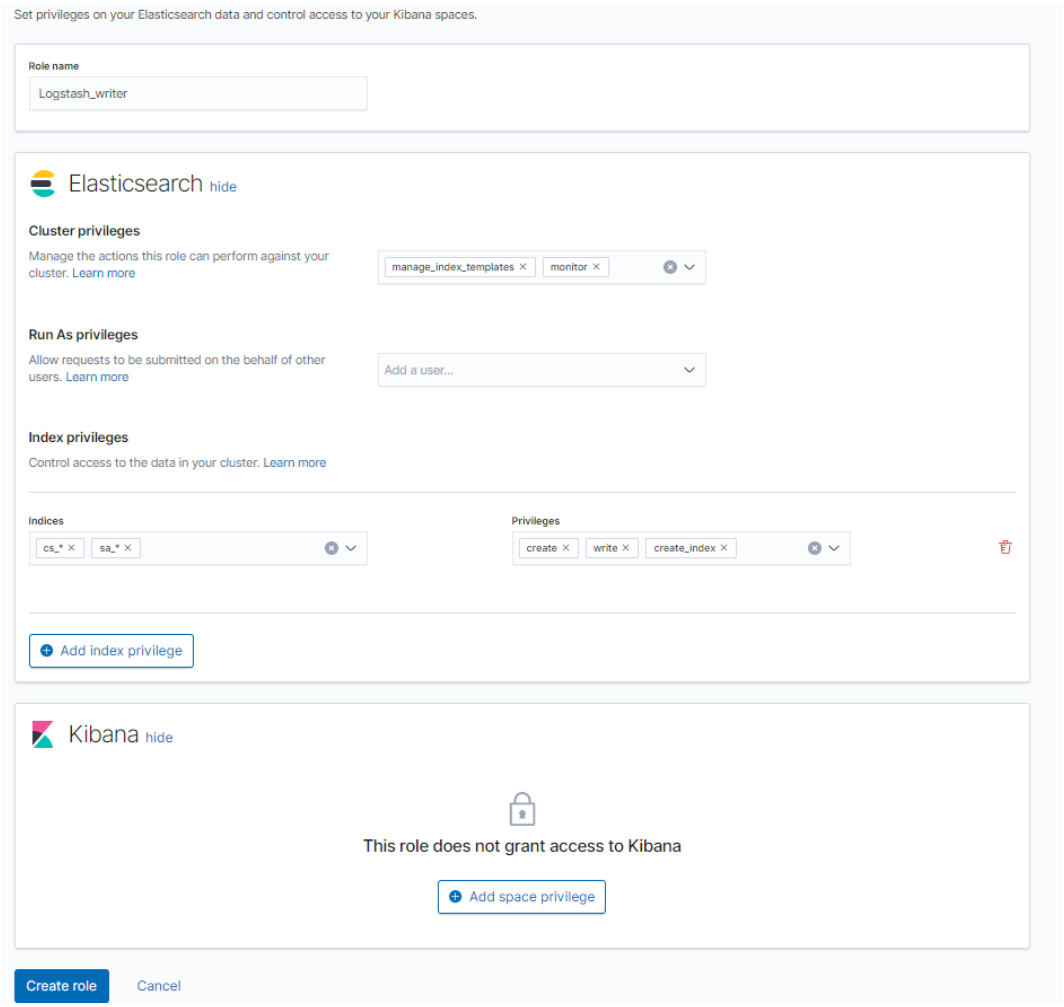


Kuvio 27. Kibanan roolien hallinta.

Valitsemalla ”create role” päästään luomaan uutta roolia. Uudelle roolille annetaan nimi ja määritellään roolin luvat ja oikeudet.

- Annetaan roolille nimeksi ”Logstash_writer”.
- ”Cluster privileges”-kohtaan asetetaan ”manage_index_templates” ja ”monitor”.
- ”Run As privileges” jätetään tyhjäksi.
- ”Index privileges” kohdassa ”Indices”-kohdalle asetetaan indeksit joihin Logstashilla on pääsy, tähän voidaan määritellä useita indeksejä tai kaikki. Määrittelyssä voidaan käyttää merkkiä ”*” villikorttina. Tällä voidaan kertoa useampi indeksi yhdellä merkinnällä, tässä työssä kaikki indeksit, joita logstash kirjoittaa alkavat joko kirjaimilla ”cs_” tai ”sa_” riippuen siitä mistä rivit on kerätty. Jos kirjoitetaan ”cs_*” ja ”sa_*” tai pelkkä ”*” saadaan kaikki mahdolliset indeksit valittua.
- privileges kohtaan valitaan ”create”, ”write” ja ”create_index”.
- ”Logstash_writer”-rooli ei tarvitse oikeuksia Kibanaan, joten se jätetään tyhjäksi.

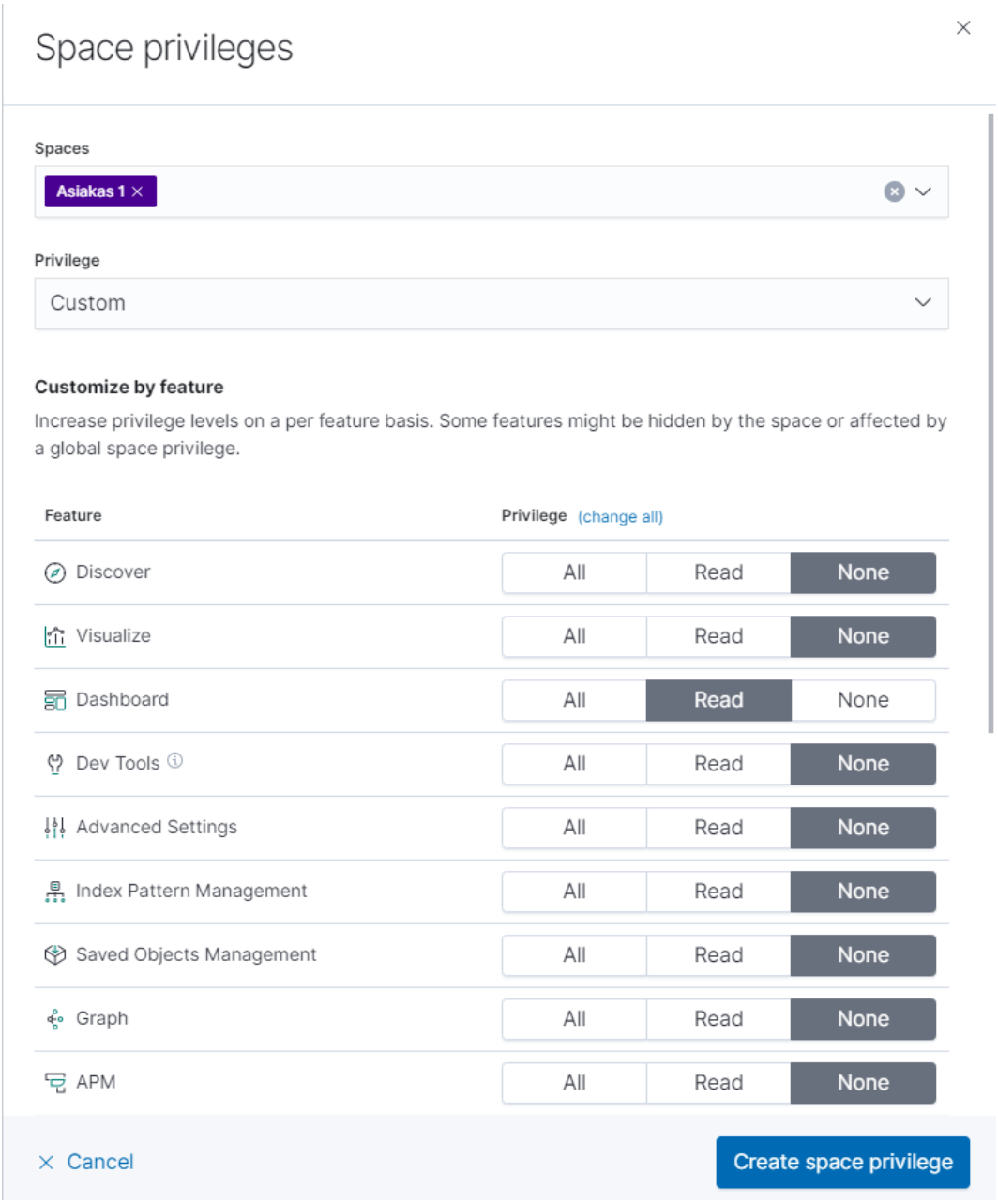
Lopuksi valitaan ”create role” **(Kuvio 28.)**. /29/



Kuvio 28. Kibana-roolin luominen.

Luodaan vielä kaksi roolia lisää aloittamalla samalla tavalla mutta annetaan ensimmäiselle näistä rooleista nimeksi ”asiakas_tila_rooli” ja toiselle nimeksi ”asiakas_tila_rooli2”. Jätetään ”cluster privileges”-kohta tyhjäksi ja ”Indices”-kohtaan syötetään ”cs_*” ja ”privileges”-kohtaan valitaan ”read”.

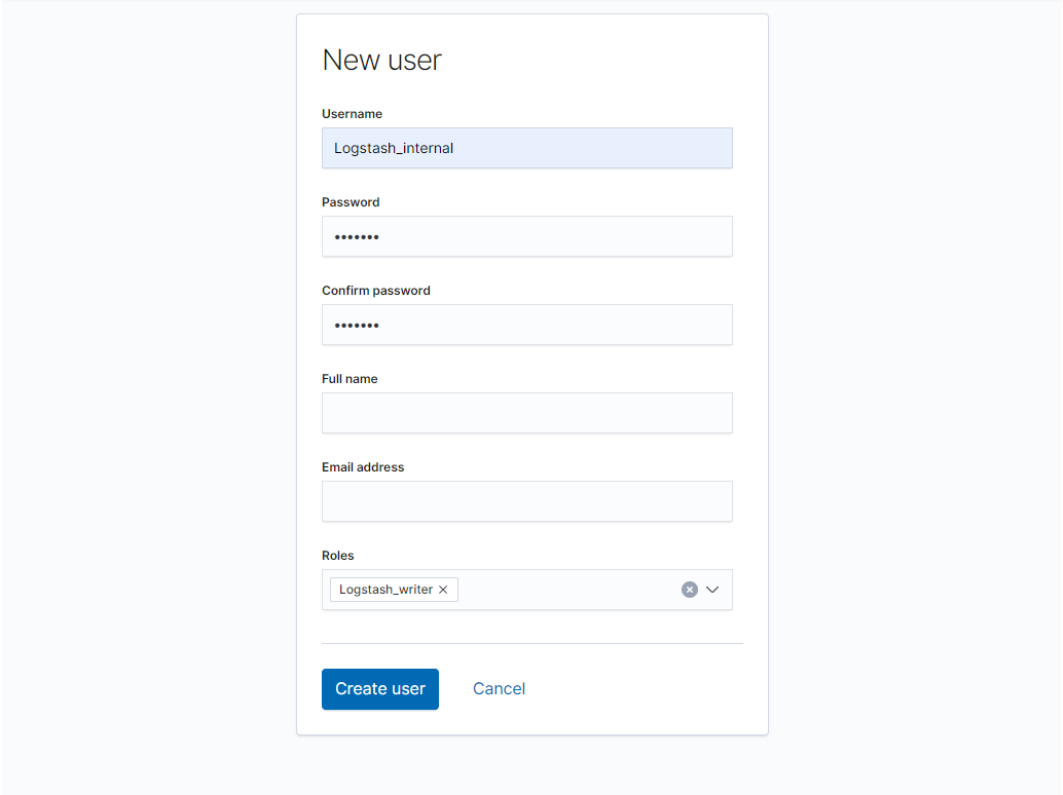
Alempana valitaan ”Add space privilege” jolloin aukeaa uusi sivuvalikko. Valikon ylimmästä osasta voidaan valita tila, johon pääsy annetaan. ”asiakas_tila_rooli”-roolin teossa valitaan ”asiakas 1”-tila ja ”asiakas-tila-rooli2”-roolin luonnissa valitaan ”asiakas 2”-tila. Valitaan ”Dashboard” kohdalle ”Read”-oikeus eli luku. Valitaan ”Create space privilege” **(Kuvio 29.)**. Lopuksi valitaan ”Create role”.



Kuvio 29. Asiakas-roolin Kibana-tilojen oikeuksien määrittely.

Kun roolit on luotu, luodaan Logstashille käyttäjä. Valitaan asetussivulla ”Security” -kohdassa ”Users”, jolloin päästään hallitsemaan ja luomaan käyttäjiä. Kuten rooleja myöskin käyttäjiä on valmiiksi sisäänrakennettu ELK-pinoon ja ne liittyvät valmiisiin ominaisuuksiin.

Valitaan ”Create user” jolloin päästään luomaan uutta käyttäjää. Annetaan uudelle käyttäjälle nimeksi esimerkiksi ”Logstash_internal” ja määritellään salasana, nimiä tai sähköposteja ei ole pakollista täyttää. Lopuksi valitaan uudelle käyttäjälle roolit, kyseisen käyttäjän kohdalla määritellään rooliksi aiemmin luotu ”Logstash_writer”. Lopuksi valitaan ”Create user” **(Kuvio 30.)**. ”Logstash_internal”-käyttäjää käytetään, kun dataa lähetetään Logstashilla Elasticsearchiin kuten määritelty luvussa 4.3 **(Kuvio 18.)**.



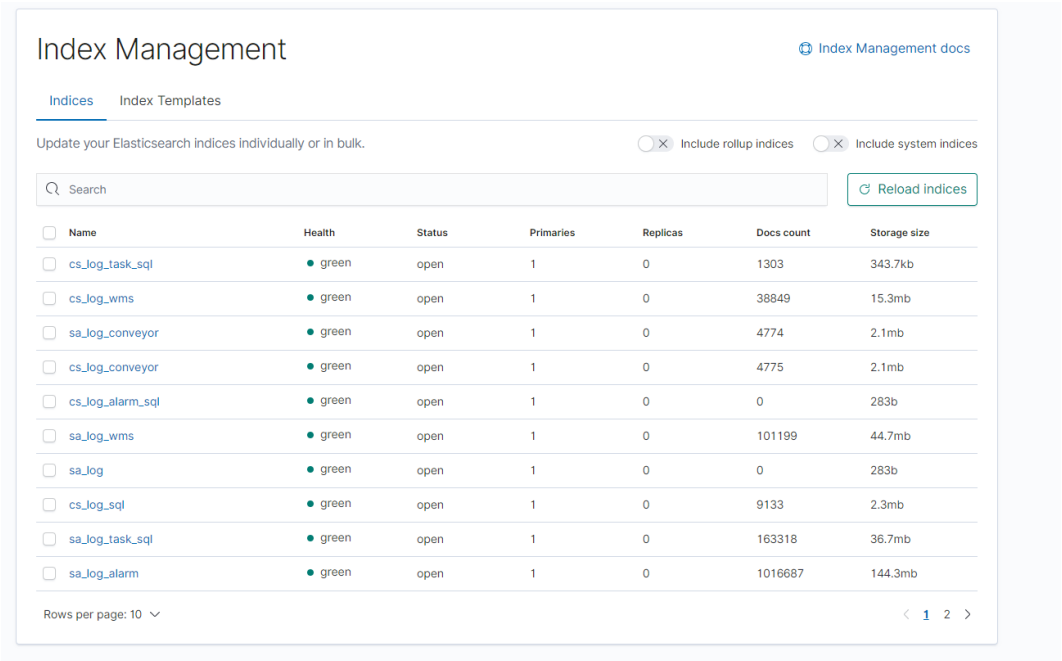
Kuvio 30. Kibana-käyttäjän luominen.

Luodaan samalla tavalla aloittaen kolme muuta käyttäjää. Nimetään käyttäjät ”asiakas”, ”asiakas2” ja ”pesmel”. ”asiakas”-käyttäjälle määritellään rooliksi aiemmin luotu ”asiakas_tila_rooli” ja ”asiakas2”-käyttäjälle määritellään rooliksi myöskin aiemmin luotu ”asiakas_tila_rooli2”. ”pesmel”-käyttäjälle määritellään rooliksi

Kibanan sisäänrakennettu ”superuser”-rooli, joka antaa käyttäjälle ”järjestelmävalvoja”-oikeudet eli täyden pääsyn ja muokkausoikeuden kaikkeen jolloin ”elastic”-käyttäjää ei tarvitse enää käyttää Kibanan hallitsemiseen.

6.3 Indeksimalli

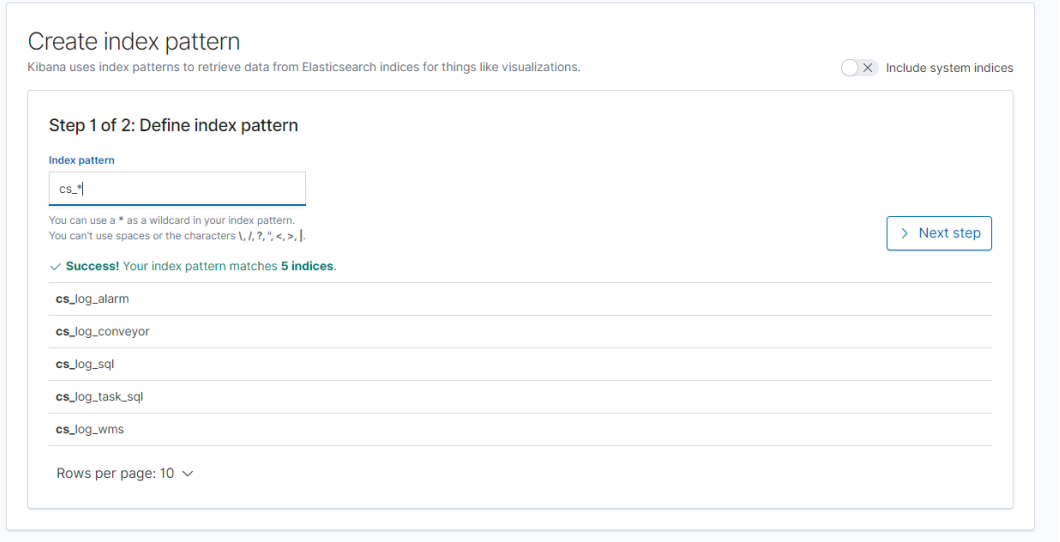
Kibana käyttää indeksimalleja päästäkseen Elasticsearchin klusterissa oleviin indeksien tietoihin ja hakeakseen tietoa niistä. Indeksit luodaan Elasticsearchiin manuaalisesti tai antamalla ”Logstash_writer”-käyttäjälle oikeudet luoda indeksejä. Kun Logstash käynnistetään ensimmäisen kerran konfiguraatiolla, johon määriteltyä indeksiä ei löydy, luo Logstash sellaisen, jos käyttäjällä on siihen oikeudet. Kun indeksi on luotu, sille voidaan Kibanassa luoda indeksimalli, jota Kibana käyttää. Kibanan kautta voidaan myös tarkastella indeksejä, jotka ovat klusterissa **(Kuvio 31.)**.



Kuvio 31. Elasticsearchissa olevia indeksejä.

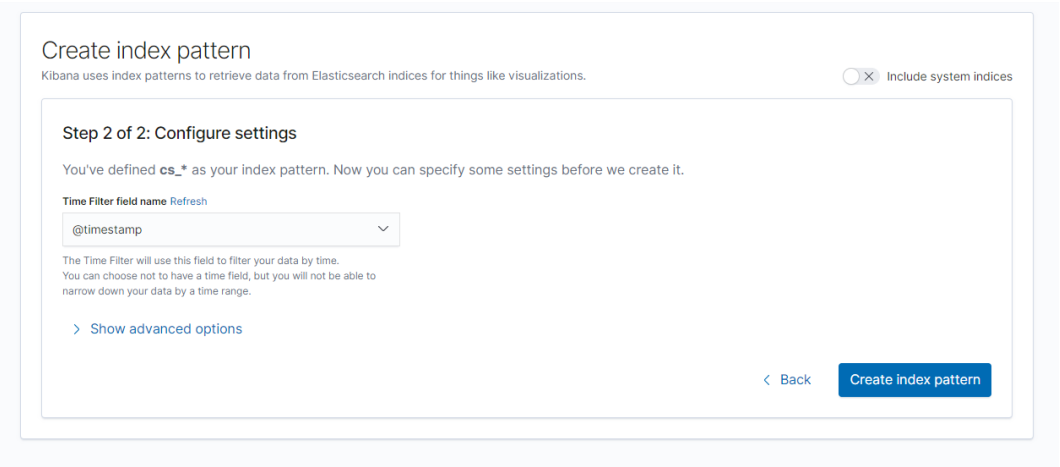
Indeksimalli luodaan Kibanan asetuksissa kohdassa ”Index patterns”. Siellä valitsemalla ”Create index pattern”, josta aukeaa uusi ikkuna missä uusi indeksimalli

määritellään. Indeksimalli määritellään kirjoittamalla sille nimi, joka vastaa yhtä tai useampaa indeksiä Elasticsearchissa. Esimerkiksi jos indeksejä on nimeltään ”cs_log_alarm” ja ”cs_log_sql” niin määrittelemällä indeksimallin nimeksi ”cs_*” voidaan tehdä indeksimalli, joka sisältää molempien indeksien tiedot **(Kuvio 32.)**. Näin voidaan yhdistellä dataa monestakin indeksistä. Jos indeksimallin nimeksi määritellään ”cs_log_sql” sisältää se vain kyseisen samannimisen indeksin tiedot.



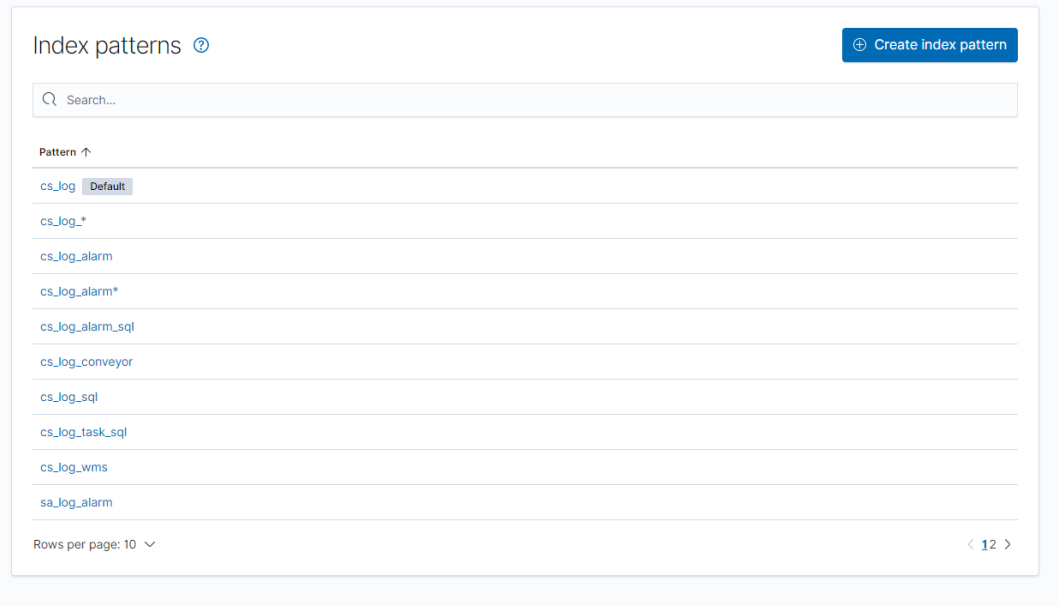
Kuvio 32. Kibanan indeksimallin indeksien määrittely.

Seuraavaksi valitaan ”next step” ja päästään seuraavalle sivulle indeksimallin luomisessa, seuraavalla sivulla valitaan kenttä, jota käytetään aikasuodatuksessa. Aiemmin luvussa 4.2 mainittu, että kerättyjen rivien aikaleimat on täsmäytetty Logstashin metadatakenttään ”@timestamp” ja tämä kyseinen kenttä valitaan indeksimallia luodessa aikasuodatuksen kentäksi **(Kuvio 33.)**. Lisäasetuksissa voidaan määritellä oma valinnainen tunnus indeksimallille. Tämä on hyödyllistä tehdä mahdollisen käyttäjävirheen tai laiterikon varalta. Kaikki visualisoinnit sidotaan tähän tunnukseen ja palauttaminen on helpompaa, mikäli tunnus on tiedossa ja helppolukuinen. Lopuksi valitaan ”Create index pattern”.



Kuvio 33. Kibanan indeksimallin aikasuodattimen määrittely.

Luodaan kaikille Logstashin luomille indekseille omat indeksimallit edellä mainitulla tyylillä **(Kuvio 34.)**. Yhdistettyjä malleja myös testattiin, mutta varsinaista käyttöä ei niille tässä tapauksessa löytynyt.



Kuvio 34. Kibanan indeksimalleja.

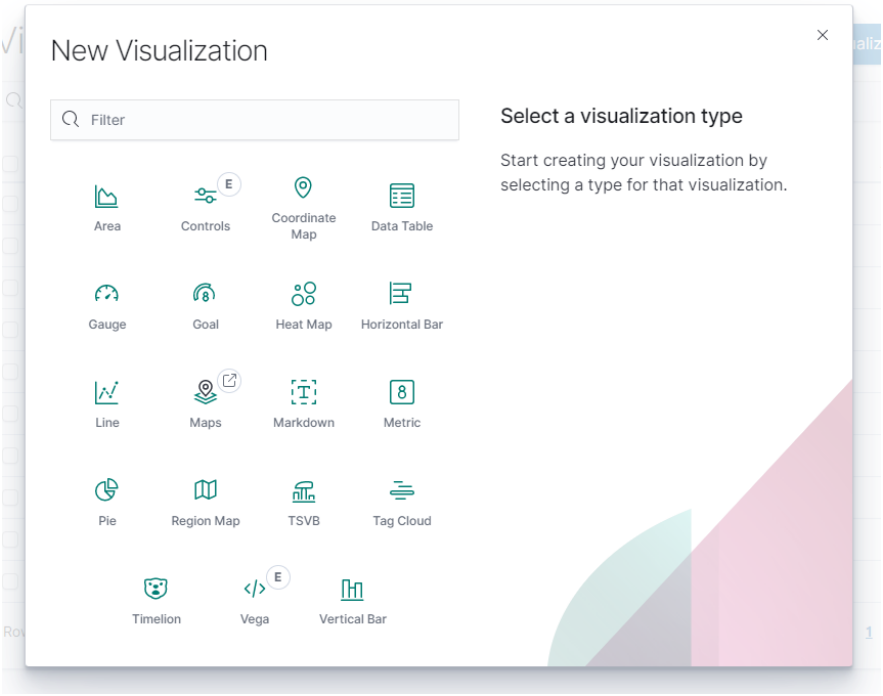
6.4 Visualisointi ja näkymät

Kibanaan luotujen indeksimallien päälle voidaan luoda visualisointeja ja visualisointeja voidaan kasata yhteen näkymiksi. Visualisoinnit tehdään aina tietylle indeksimallille, mutta se on mahdollista muuttaa luomisen jälkeen manuaalisesti tarkastelemalla visualisoinnin JSON-lähdekoodia asetuksissa ja korvaamalla referenssikentän ”id”-kenttä toisen indeksimallin tunnusta vastaavaksi. Tällaisessa operaatiossa on hyödyllistä, jos indeksimallille on määritelty valinnainen uniikki tunnus sitä luodessa kuten mainittu luvussa 6.3, koska Kibana luo sen muuten automaattisesti satunnaiseksi uniikiksi merkkijonoksi. Visualisointeja voi myös viedä toiseen projektiin tai tuoda toisesta projektista.

6.4.1 Visualisointien luominen

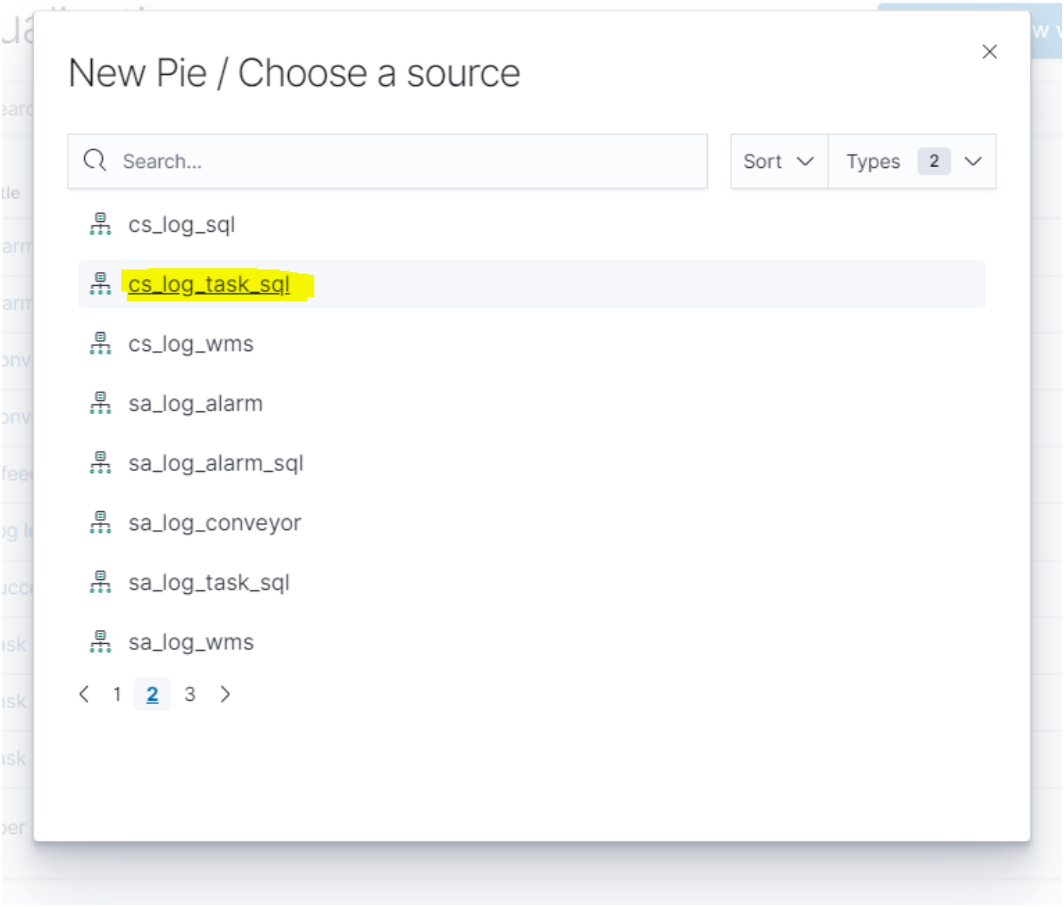
Visualisoinnit luodaan siihen tilaan, joka on Kibanassa valittuna. Kuten luvussa 6.1 on selitetty, on tähän työhön jo luotu kaksi erillistä tilaa asiakkaille oletustilan lisäksi. Mikäli käyttäjällä on siihen oikeudet, voidaan tilaa vaihtaa Kibanassa vasemmasta yläreunasta valitsemalla nykyisen tilan kuvake ja valitsemalla mahdollinen toinen tila. Valitaan esimerkiksi tilaksi ”asiakas 1”.

Visualisointeja pääsee luomaan Kibanan käyttöliittymässä vasemmasta reunasta kohdasta ”visualize” ja Kibanaan aukeaa lista kaikista visualisoinneista. Uusi visualisointi voidaan luoda valitsemalla ”Create new visualization”. Kibanaan voi luoda usean tyyppisiä visualisointeja esimerkiksi piirakkakaavioita, pylväsdiagrammeja, viivakaavioita, tauluja ja mittareita **(Kuvio 35.)**.



Kuvio 35. Kibanan visualisointityypit.

Luodaan piirakkakaavio valitsemalla tyypiksi ”pie”, tyypin valitsemisen jälkeen valitaan indeksimalli, johon visualisointi sidotaan. Valitaan tässä esimerkissä indeksimalliksi ”cs_log_task_sql” **(Kuvio 36)**, joka sisältää dataa tietokantataulusta, johon on kerätty tiedot suoritetuista tehtävistä. Kerätty data on luotu WMS-järjestelmän simulaatiossa.

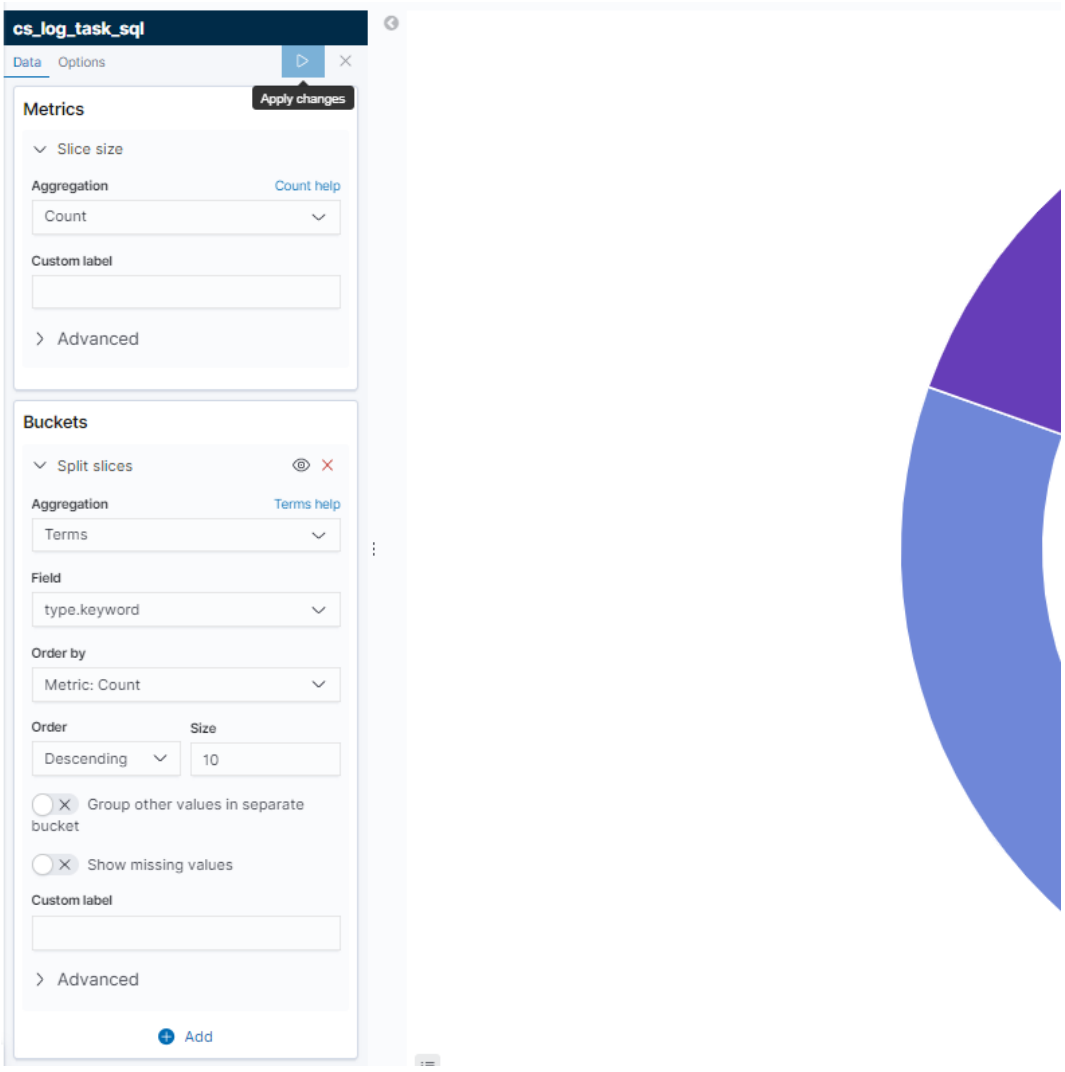


Kuvio 36. Kibanan visualisoinnin indeksimallin valinta.

Luodaan piirakkakaavio, jossa nähdään aikavälillä määrä, montako erityyppistä tehtävää on suoritettu. Visualisoinnin luomistyökalussa vasemmalla on konfiguraatiolaatikot, joilla määritellään mitä visualisoinnin tahdotaan näyttävän. Ylempi ”Metrics”-laatikko voidaan jättää oletukseensa, ”Metrics”-laatikon ”Aggregation” on aina oletuksena ”Count” eli laskenta jos se on mahdollista. Seuraavaksi lisätään sankoja eli ”Buckets”-laatikkoon lisätään uusi määrittely valitsemalla ”Add”. Ylimmäksi valitaan ”Split slices” eli piirakan alueet jaetaan erilleen. ”Aggregation”-kohdalle valitaan ”Terms” eli termit ja ”Field” eli indeksimallin kentäksi ”type.keyword”.

Tärkeää on huomata, että kenttä täytyy olla ”.keyword”-loppuinen, koska kyseiset kentät on säilytetty yhtenä eikä niitä ole rikottu erillisiksi sanoiksi indeksissä. Tällaisille kentille voidaan tehdä ryhmittelyjä, jota tällaisessa visualisoinnissa ollaan tekemässä. ”.keyword”-loppuiset kentät ovat SQL-kielessä verrattavissa ”GROUP BY”-lauseisiin.

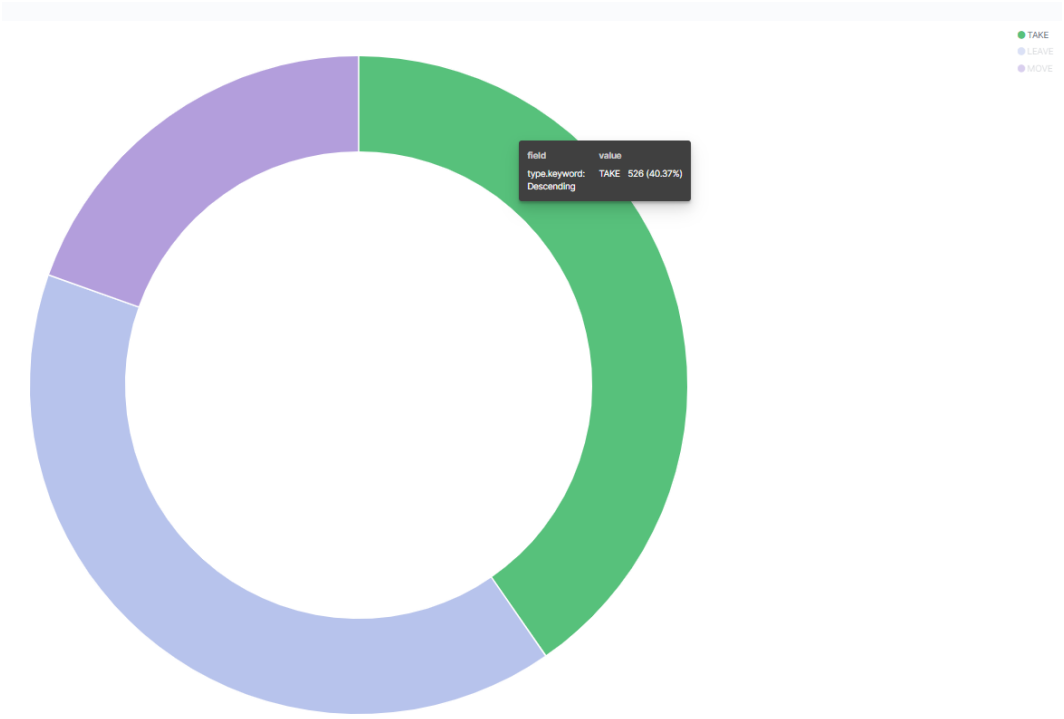
Näin saadaan kaikki ”type.keyword”-kentässä esiintyvät termit lisättyä omiin sankoihinsa ja laskettua niiden esiintymät yhteen. Lopuksi valitaan ”order”-kohdassa järjestys, jossa sangot esitetään. ”size”-kentällä voidaan rajoittaa, montako termiä esitetään. Esimerkiksi jos ”order” on ”Descending” ja ”Size” on 10 saadaan esiintymien yhteenlasketun määrän mukaan laskevassa järjestyksessä kymmenen termiä **(Kuvio 37.)**.



Kuvio 37. Visualisoinnin konfigurointi.

Ennestään on tiedossa, että erilaisia termejä on vain kolme, mutta voidaan laittaa ”Size”-arvoksi 10. Tarkasteltavaa aikaväliä voidaan dynaamisesti vaihtaa oikeassa yläkulmassa olevan aikasuodattimen avulla.

Nyt nähtävillä on piirakkakaavio, josta erottuvat kolme erilaista tehtävätyyppiä, niiden suhteet ja niiden määrät liikuttamalla hiirtä piirakan siivujen kohdille **(Kuvio 38.)**. Lopuksi valmis visualisointi voidaan tallentaa valitsemalla ”Save” ja antamalla sille nimi.



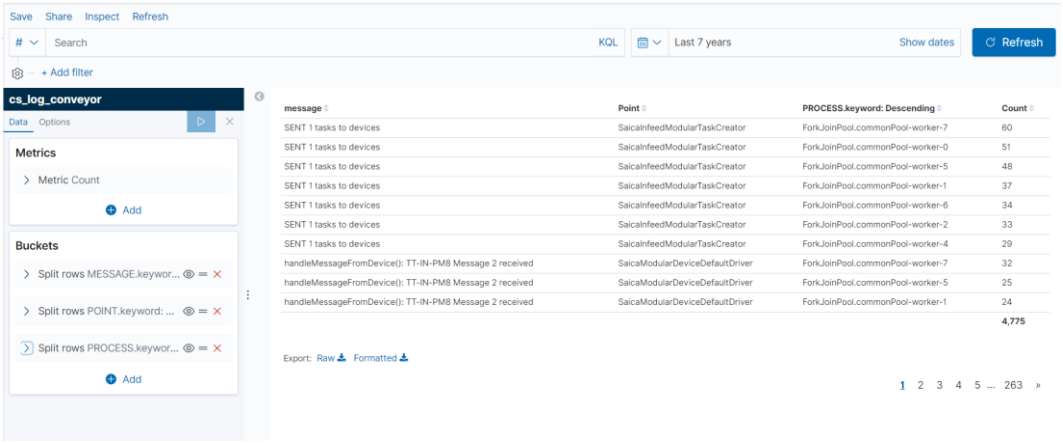
Kuvio 38. Kibanan piirakkakaavio visualisointi.

Visualisointi voidaan luoda myös taulukon muodossa **(Kuvio 39.)**, esimerkiksi vaikeasti luettavat ja haettavat lokitiedostot saadaan muokattua siistiksi tauluksi, jossa voidaan laskea vaikka tiettyjen tapahtumien esiintymiä aikavälillä.

Taulukonmuotoinen visualisointi on hyödyllinen varsinkin lokitiedostojen visualisoinnissa, koska Kibana mahdollistaa vielä erillisen haun tekemisen jo valmiista visualisoinnista tai näkymästä. Tällöin voidaan hakea tiettyjä tapahtumia, tietyllä aikavälillä lokitiedostoista ja saada ne siististi taulukon muodossa. Taulukko voidaan yhdistää näkymällä esimerkiksi pylväsdiagrammeihin. Tällöin on mahdollista saada helposti ja nopeasti kuva siitä, mitä WMS-järjestelmässä on tapahtunut tietyllä hetkellä tai onko siellä tapahtunut jotain tiettyä.

Ongelmaksi alkaa suurilla datamäärillä muodostua ”sankojen” eli ryhmien määrän kasvaminen, varsinkin lokitiedostoissa erilaisia viestejä tulee paljon ja jos lokeja

tarkastellaan pitkällä aikavälillä, tulee ryhmien enimmäismäärä täyteen. Oletuksena ryhmiä voi olla enintään 65536-kappaletta. /30/

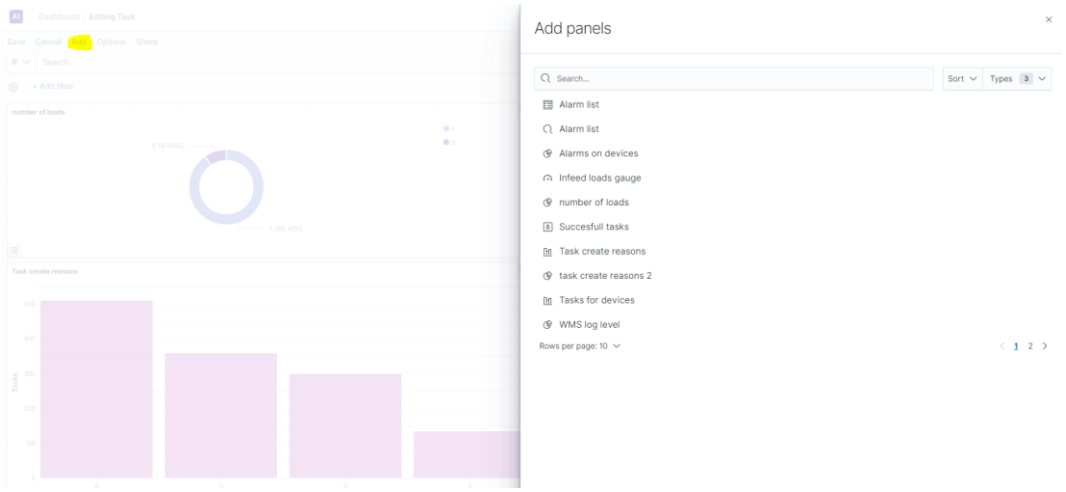


Kuvio 39. Kibanan taulukkomuotoinen visualisointi.

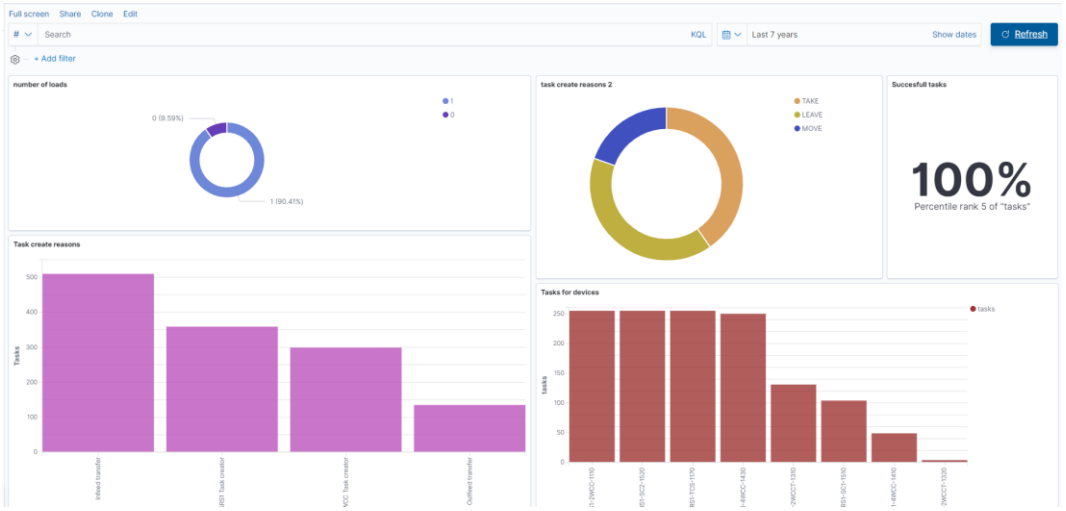
6.4.2 Näkymien luominen

Samalla tavalla kuin visualisoinnit myös näkymät luodaan siihen tilaan, joka Kibanassa on kyseisellä hetkellä valittuna. Kun visualisointeja halutaan esittää yksi tai useampi samalla ruudulla täytyy niistä luoda näkymä, näkymässä voi myöskin tehdä hakuja käyttäen KQL-kieltä ja tutkia visualisointien tarjoaman tiedon kytköksiä **(Kuvio 42.)**.

Näkymiä pääsee luomaan Kibanassa, valitsemalla vasemmalta sivupalkista ”Dashboard” ja valitsemalla ”Create new dashboard”. Luodaan uusi näkymä, johon kerätään WMS-tehtäviin liittyviä visualisointeja. Valitsemalla oikeasta yläreunasta ”add” saadaan auki valikko, josta tilaan luotuja visualisointeja voidaan valita näkymään. Kun visualisointeja valitaan listasta, lisätään ne näkymään **(Kuvio 40.)**. Visualisointeja voidaan liikutella ja niiden kokoa voidaan muuttaa näkymässä hyvin vapaasti käyttäjän mielen mukaan **(Kuvio 41.)**.

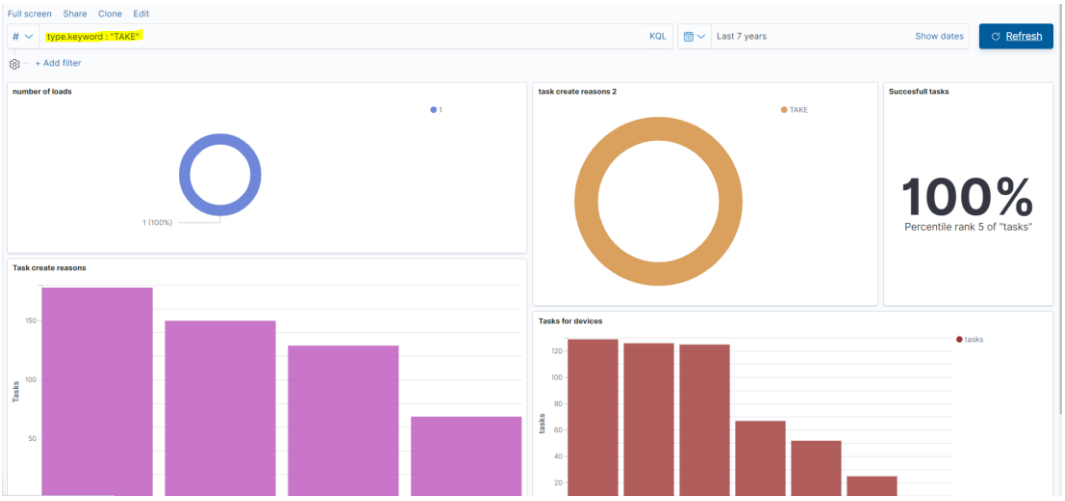


Kuvio 40. Kibanan näkymän luominen.



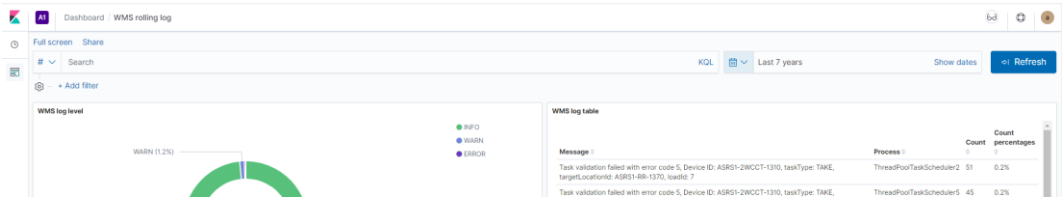
Kuvio 41. Visualisointien asetteleminen näkymässä.

Kibanan näkymästä voidaan tehdä vapaita hakuja kirjoittamalla Kibanan yläpalkkiin KQL-kielellä kysely. Esimerkiksi Kuviossa 41 näkyy ”task create reasons 2”-paneeli, jossa on esillä mitä tehtävätyyppejä on aikavälillä tapahtunut. Jos yläosan hakupalkkiin kirjoitetaan kyselyksi esimerkiksi ”type.keyword : "TAKE"” saadaan esille vain niihin tehtäviin liittyvät tiedot, joiden tyyppi on ”TAKE” **(Kuvio 42.)**.



Kuvio 42. Haku Kibanan näkymästä.

Kun kirjaudutaan Kibanaan ”asiakas”-tunnuksilla päästään vain katselemaan ”asiakas 1”-tilassa olevia näkymiä, kuten luvussa 6.2 ”asiakas”-käyttäjälle asetettiin **(Kuvio 43.)**.



Kuvio 43. Asiakkaan näkymä.

7 LOPPUPÄÄTELMÄT

Kokonaisuutena ELK-pino on erittäin joustava ja tilanteeseen muokattavissa oleva alusta. Ominaisuuksiensa ja testauksen puolesta ELK-pino soveltuu erittäin hyvin määriteltyihin tarpeisiin. Lisäkehityksellä Kibanaan voisi myös rakentaa monimutkaisempia työpöytiä ja kyselyitä, jotka toimisivat vielä paremmin vianselvityksessä ja asiakkaan tiedottamisessa.

ELK-pinon puolesta puhuu monikäyttöisyys ja keveys. Järjestelmän kuormitus tuskin on ongelma nykystandardeilla, mutta tallennustilaa järjestelmä vaatii tietenkin sen mukaisesti kuin kerättäviä rivejä tulee. Testeissä noin miljoona kerättyä riviä tietoa vaati noin 150-megabittiä tilaa, miljoona riviä on kuitenkin melko vaatimaton määrä. Seuraavaksi ELK-pinoa olisi mielenkiintoista testata jossain oikeassa projektissa, jolloin näkisi millaisella tahdilla Elasticsearch alkaisi kasvaa kooltaan, kun lokirivejä generoituu paljon nopeammin ja enemmän. ELK-pino on myös melko nopea siirtää ja konfiguroida jokaiseen projektiin erikseen. Rivejä voidaan myöskin kerätä lähes jokaisesta mahdollisesta lähteestä.

Mahdollinen pilvipalvelu voisi olla myöskin yksi toteutusvaihtoehto, jossa Pesmel tarjoaa asiakkailleen pilvipalvelun, johon kerätään usean asiakkaan data ja jokainen asiakas pääsee katselemaan vain omaa dataansa. Tällaisessa tapauksessa tallennustilan riittävyys olisi palveluntarjoajalla. Helpoin ratkaisu todennäköisesti olisi tarjota palvelua asiakkaalle paikallisesti esimerkiksi heidän serverillään.

Ongelmaksi tässä tapauksessa muodostuu se, että ELK-pinon ilmaisversio ei salli palvelumuotoisen kokonaisuuden tarjoamista asiakkaalle. Pesmel saisi kyllä tarjota palvelua, jossa asiakkaalle pystytetään ELK-pino heidän järjestelmäänsä, mutta sitä ei saisi tarjota palvelumuotoisena, jossa asiakas maksaa esimerkiksi kuukaudessa kiinteän hinnan ELK-pinon käytöstä. /31/

Mikäli haluttaisiin tarjota palvelumuotoista ympäristöä asiakkaalle, jossa Pesmel olisi palveluntarjoaja ja asiakas saisi Kibanaan pääsyn, tarvittaisiin ensin neuvotella Elastic B.V:n kanssa lisenssimaksuista ELK-pinon käytöstä.

LÄHTEET

/1/ Pesmel Oy. Kauhajoki. Yhteystiedot. Viitattu 5.3.2021. https://www.finder.fi/Metallituotteet/Pesmel+Oy/Kauhajoki/yhteystiedot/1933081?ref=redirect

/2/ Pesmel. About us. Viitattu 5.3.2021. https://pesmel.com/about-us/

/3/ Elastic. What is. ELK stack. Viitattu 5.3.2021. https://www.elastic.co/what-is/elk-stack

/4/ Elastic. Logstash. Viitattu 6.3.2021. https://www.elastic.co/logstash

/5/ Elastic. Beats. Viitattu 6.3.2021. https://www.elastic.co/beats/

/6/ Elastic. Logstash. Configuration. Viitattu 6.3.2021. https://www.elastic.co/guide/en/logstash/current/configuration.html

/7/ Elastic. Logstash. Configuration file structure. Viitattu 6.3.2021. https://www.elastic.co/guide/en/logstash/current/configuration-file-structure.html

/8/ Elastic. Elasticsearch. Viitattu 8.3.2021. https://www.elastic.co/what-is/elasticsearch

/9/ Elastic. Elasticsearch. Settings. Viitattu 8.3.2021. https://www.elastic.co/guide/en/elasticsearch/reference/current/settings.html

/10/ Elastic. Kibana. Viitattu 8.3.2021. https://www.elastic.co/what-is/kibana

/11/ Elastic. Kibana. Kuery query. Viitattu 9.3.2021. https://www.elastic.co/guide/en/kibana/current/kuery-query.html

/12/ Elastic. Logstash. Input plugins. Viitattu 9.3.2021. https://www.elastic.co/guide/en/logstash/current/input-plugins.html

/13/ Elastic. Downloads. Elasticsearch. Viitattu 14.3.2021. https://www.elastic.co/downloads/elasticsearch

/14/ Elastic. Cloud. Viitattu 16.3.2021. https://www.elastic.co/cloud/

/15/ Elastic. Downloads. Logstash. Viitattu 16.3.2021. https://www.elastic.co/downloads/logstash

/16/ Elastic. Downloads. Kibana. Viitattu 17.3.2021. https://www.elastic.co/downloads/kibana

/17/ Elastic. Logstash. Multiple pipelines Viitattu 22.3.2021. https://www.elastic.co/guide/en/logstash/current/multiple-pipelines.html

/18/ Microsoft. Download microsoft jdbc driver. Viitattu 25.3.2021. https://docs.microsoft.com/en-us/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server?view=sql-server-ver15

/19/ Elastic. Logstash. Plugin inputs JDBC. Viitattu 25.3.2021. https://www.elastic.co/guide/en/logstash/current/plugins-inputs-jdbc.html

/20/ Elastic. Logstash. Plugin inputs file. Viitattu 25.3.2021. https://www.elastic.co/guide/en/logstash/current/plugins-inputs-file.html

/21/ Elastic. Logstash. Plugin codecs multiline. Viitattu 1.4.2021. https://www.elastic.co/guide/en/logstash/current/plugins-codecs-multiline.html

/22/ Elastic. Logstash. Filter plugins. Viitattu 1.4.2021. https://www.elastic.co/guide/en/logstash/current/filter-plugins.html

/23/ Elastic. Logstash. Plugin filters Grok. Viitattu 1.4.2021. https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html

/24/ Elastic. Logstash. Plugin filters mutate. Viitattu 5.4.2021. https://www.elastic.co/guide/en/logstash/current/plugins-filters-mutate.html

/25/ Elastic. Logstash. Output plugins. Viitattu 10.4.2021. https://www.elastic.co/guide/en/logstash/current/output-plugins.html

/26/ Elastic. Subscriptions. Viitattu 10.4.2021. https://www.elastic.co/subscriptions

/27/ Elastic. Logstash. Tuning Logstash. Viitattu 11.4.2021. https://www.elastic.co/guide/en/logstash/current/tuning-logstash.html

/28/ Elastic. Reference. Security minimal setup. Viitattu 18.4.2021. https://www.elastic.co/guide/en/elasticsearch/reference/7.x//security-minimal-setup.html

/29/ Elastic. Logstash. Logstash security. Viitattu 5.5.2021 https://www.elastic.co/guide/en/logstash/current/ls-security.html

/30/ Elastic. Elasticsearch. Search aggregations bucket. Viitattu 5.5.2021. https://www.elastic.co/guide/en/elasticsearch/reference/master/search-aggregations-bucket.html

/31/ Elastic. Elastic license. Viitattu 7.5.2021. https://www.elastic.co/licensing/elastic-license