Opinnäytetyö AMK

Tieto- ja viestintätekniikka

2019

Tuomo Pihlasto

ELASTIC STACKIN HYÖDYNTÄMINEN NEXTGEN-INTEGRAATIO-OHJELMISTON VALVONNASSA

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ AMK | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka

2019 | 34 sivua

Tuomo Pihlasto

ELASTIC STACKIN HYÖDYNTÄMINEN NEXTGEN-INTEGRAATIO-OHJELMISTON VALVONNASSA

NextGen Connect -integraatio-ohjelmiston käyttöliittymästä ei ole helppoa seurata lokiviestejä ja käyttöliittymä vaatii erillisen kirjautumisen valvottavaan palvelimeen. Virheiden sattuessa usein ei tiedetä, mikä virheen on aiheuttanut, jolloin etsitään lokiviesteistä lisätietoa. Integraatioratkaisujen aiheuttamia virheitä eikä lokeja tarkkailla aktiivisesti, vaan niihin reagoidaan, jos ne huomataan.

Työn tavoitteena oli hyödyntää AgentIT:llä jo käytössä olevaa Elastic Stackia NextGen Connect -integraatio-ohjelmiston valvomiseen. Toteutetusta valvontakokonaisuudesta tuli valmistaa hyvin dokumentoitu kokonaisuus, joka olisi toteutettavissa uudelleen useammille asiakkaiden palvelimille, joissa NextGen Connect -integraatio-ohjelmisto on käytössä.

Opinnäytetyössä tutkittiin työssä käytettävien Elastic Stack -komponenttien teoriaa ja sitä, miten niitä mahdollisesti voitaisiin hyödyntää työssä. Lisäksi työssä perehdyttiin AgentIT:n käyttämään Grafana-ohjelmaan ja siihen, mitä kaikkea sillä olisi mahdollista tehdä ja mitä AgentIT:llä ei hyödynnetä.

Käytännön osuudessa rakennettiin ensin erilliseen testiympäristöön todellisuutta kuvaava ympäristö, johon kokonaisuus tultaisiin asentamaan. Testiympäristössä Elastic Stackin komponentteihin tehtiin konfiguraatiomuutokset, jotka saivat aikaan halutun tuloksen. Grafanaan tehtiin valmiit visualisoinnit, joista saatiin haluttu tieto esille ja nämä visualisoinnit olivat suoraan vietävissä tuotannon ympäristöön.

ASIASANAT:

Elastic Stack, Elasticsearch, Grafana, NextGen, valvonta, loki-tiedostot

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and communications technology

2019 | 34 pages

Tuomo Pihlasto

USING ELASTIC STACK IN MONITORING NEXTGEN INTEGRATION ENGINE

Using the NextGen Connect integration engine's user interface for checking logs is not intuitive and it requires user to separately log onto the machine where the integration engine is located. When errors occur, the cause is not often clear, so the logs are checked for further info. Logs and errors caused by integrations are not actively monitored so they are only checked if something unusual is noticed.

The aim of the thesis was to deploy Elastic Stack, which is already used by AgentIT in other types of monitoring, to monitor NextGen Connect integration engine. Accomplished monitoring solution should be well documented and based on those created documentations it should be easily replicated to several servers where NextGen Connect integration engine is used.

This thesis explores the theory and possibilities of Elastic Stack components used in the thesis and how these possibilities could be used in the thesis. In addition, this thesis investigated program called Grafana, which is used by AgentIT and some of its customers, and what it can do and what AgentIT have not done or experimented yet.

The practical part of the thesis started with creating a test environment that replicated a production environment where the monitoring solution would be implemented first. Configuration changes were made to the Elastic Stack components so that the wanted information could be extracted and visualized in Grafana. Visualized graphs and panels created into the test environment could be imported into production environment by ease.

KEYWORDS:

Elastic Stack, Elasticsearch, Grafana, NextGen, monitoring, log-files

SISÄLTÖ

KUVAT

TAULUKOT

KÄYTETYT LYHENTEET

API	Application Programming Interface, rajapinta, jonka kautta ohjelmat voivat kommunikoida keskenään
HTTP	HyperText Transfer Protocol, siirtoprotokolla, jolla selainten ja web-palvelimien viestit siirtyvät
ICMP	Internet Control Message Protocol, kontrolliprotokolla, jolla lähetetään viestejä tietokoneesta toiseen
Indeksi	Ryhmä sirpaleita, jotka muodostavat varaston datalle (engl. index)
JMX	Java Manage Extensions, ohjelma, jolla kontrolloidaan javan lisäosia
JSON	JavaScript Object Notation, tiedostomuoto
JVM	Java Virtual Machine, virtuaalikone, joka pyörittää javaa
Klusteri	Ryhmä noodeja, jotka muodostavat Elasticsearch kokonaisuuden (engl. cluster)
Noodi	Yksittäinen tietokone tai java-instanssi, jolla Elasticsearch toimii (engl. node)
Regex	Regular Expression, sarja merkkejä, jotka vastaavat haettua kaavaa
Sirpale	Lucene indeksi, joka varastoi ja prosessoi osan Elasticsearch indeksistä (engl. shard)
Sisääntulo	Portti, joka on auki ja kuuntelee sekä vastaanottaa dataa (engl. input)
SSL	Secure Sockets Layer, protokolla, joka suojaa verkon yli tapahtuvaa viestintää
Syntaksi	Ohjelmointikielen hyväksymät komennot ja merkit
TCP	Transmission Control Protocol, tietoliikenneprotokolla, jolla luodaan yhteyksiä tietokoneiden välille, joilla on pääsy internetiin
TLS	Transport Layer Security, protokolla, joka suojaa verkon yli tapahtuvaa viestintää
Ulostulo	Portin osoite, johon data lähetetään (engl. output)
YAML	YAML Ain’t Markup Language, tiedostomuoto

1 JOHDANTO

Opinnäytetyön toimeksiantajana on suomalainen AgentIT Finland Oy, joka on järjestelmäintegraatioihin ja niiden valvontaan erikoistunut asiantuntijatalo. Heidän pääkonttori sijaitsee Turussa ja Helsingissä on pienempi konttori. Työntekijöitä ja harjoittelijoita on yhtiössä nyt noin 20.

Opinnäytetyön tavoitteena on suunnitella ja toteuttaa Elastic Stackia hyödyntämällä valvontakomponentti NextGen Connect -integraatio-ohjelmalle. Tällä hetkellä NextGen Connect (ent. Mirth Connect) -integraatio-ohjelman lokitiedot pitää katsoa jokaiselta koneelta erikseen ohjelman käyttöliittymän kautta. Se vaatii aina kirjautumisen kyseiselle koneelle ja vielä erikseen kirjautumisen NextGen Connect -ohjelmaan. Valvontakomponentilla tarvittavat tiedot saataisiin kerättyä jäsennetyssä muodossa AgentIT:n omalle palvelimelle kirjautumatta valvottaville palvelimille. Kerätyt tiedot visualisoidaan Grafanaa hyödyntämällä erilaisiksi valvontanäkymiksi, joista näkyy nopealla silmäyksellä valvontaa kiinnostava tieto. Loki- ja muut tiedot ovat usein vaikeita luettavia alkuperäisessä muodossaan ja siksi valvontakomponentti jäsentää tiedoista tarpeelliset kohdat omiksi osiksiin. Tiedot olisivat luettavammassa muodossa ja niihin pystytään lisäämään hälytyksiä, jos jokin tietty virhe ilmenee tai asetettu raja ylittyy.

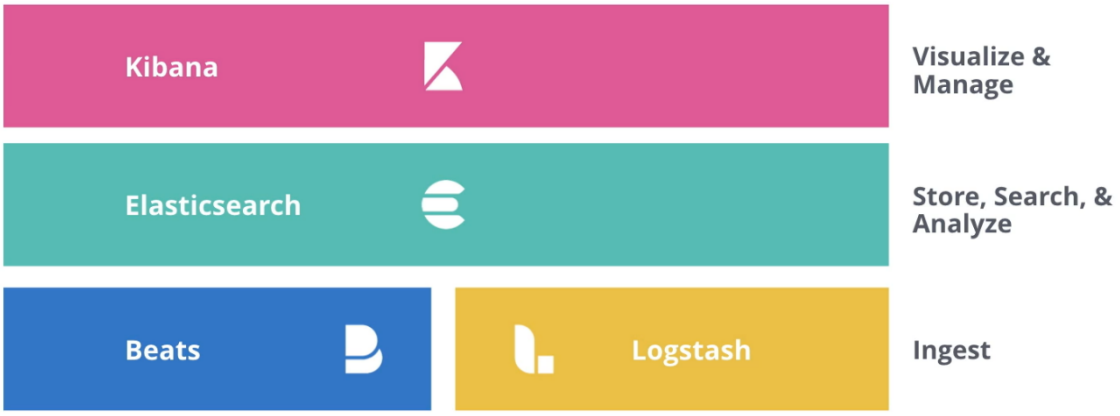
Työn tavoitteena on kehittää valmis valvontajärjestelmä, joka on helposti replikoitavissa dokumentaatioiden avulla. AgentIT:llä on jo useampi asiakas, joiden käytössä on NextGen Connect -integraatio-ohjelmisto ja joille tämä järjestelmä on helposti myytävissä. Työssä hyödynnetään Elastic Stackiin kuuluvia Elasticsearch-, Filebeat-, Metricbeat-, Heartbeat-, Logstash- ja Kibana-komponentteja sekä Grafanaa, joka ei ole osana Elastic Stackia.

Työn alkuun käydään läpi kyseisten komponenttien teoriaa, miten ne toimivat ja miten niitä voidaan käyttää. Teorian jälkeen käydään läpi, miten kyseisiä komponentteja on käytetty tässä työssä, ja näytetään, minkälaisia visuaalisointeja niiden keräämästä datasta on toteutettu.

2 ELASTIC STACK

Elastic Stack on Elastic yhtiön tarjoama kokonaisuus, joka mahdollistaa erilaisten datan keräyksen eri lähteistä sekä tämän datan etsimisen, analysoinnin ja visualisoinnin reaaliajassa (Kuva 1.). Elastic Stackiin kuuluu Elasticsearch, Beats, Logstash ja Kibana. Elasticsearch on koko kokonaisuuden keskeisin ja tärkein osa [8].

Elastic Stackiin pystyy lisäämään useita eri tiedonkerääjiä (Beats), jotka auttavat keräämään monipuolisempaa dataa. Sillä pystyy hakemaan luotettavasti ja turvallisesti dataa lähteestä ja datan muodosta riippumatta. Sen jälkeen varastoitua dataa voidaan hakea, analysoida ja visualisoida reaaliajassa. Elastic Stackiin kuuluvat komponentit ovat avointa lähdekoodia, jolloin kuka vaan voi ladata ja käyttää niitä ilmaiseksi. Elastic Stackiin kuuluvasta Kibanasta on olemassa myös maksullisia versioita, mutta niitä ei tässä työssä tulla käyttämään. Maksulliset versiot tarjoavat parempaa suojaa kerätylle datalle ja muita ominaisuuksia, kuten mahdollisuuden tehdä hälytyksiä ja jatkuvan käyttäjätuen [8].



Kuva 1. Elastic Stack [8].

NextGen Connect (ent. Mirth Connect) on alunperin terveydenhuollon tarpeisiin suunniteltu integraatioalusta, jota käytetään sanomien välittämiseen, suodattamiseen ja muuntamiseen eri järjestelmien välillä. Se perustuu avoimeen lähdekoodiin, jolloin sitä on ilmaista käyttää. Siitä on myös maksullinen versio, jossa tulee perusohjelmiston lisäksi erilaisia lisäosia sekä täysi käyttäjätuki [13].

2.1 Elasticsearch

Elasticseach kerää eri Elastic Stackin komponenttien keräämän datan varastoon, josta sitä voidaan analysoida, hakea ja visualisoida. Elasticsearch varastoi datansa yhteen tai useampaan sen indekseistä. Vaikka indeksillä on yleisesti ohjelmoinnissa eri merkitys, voidaan Elasticsearchin tapauksessa indeksejä ajatella tietokantana eli varastona, johon on tallennettu tietoa. Elasticsearch kerää ja tallentaa dokumentit JSON-formaatissa [8].

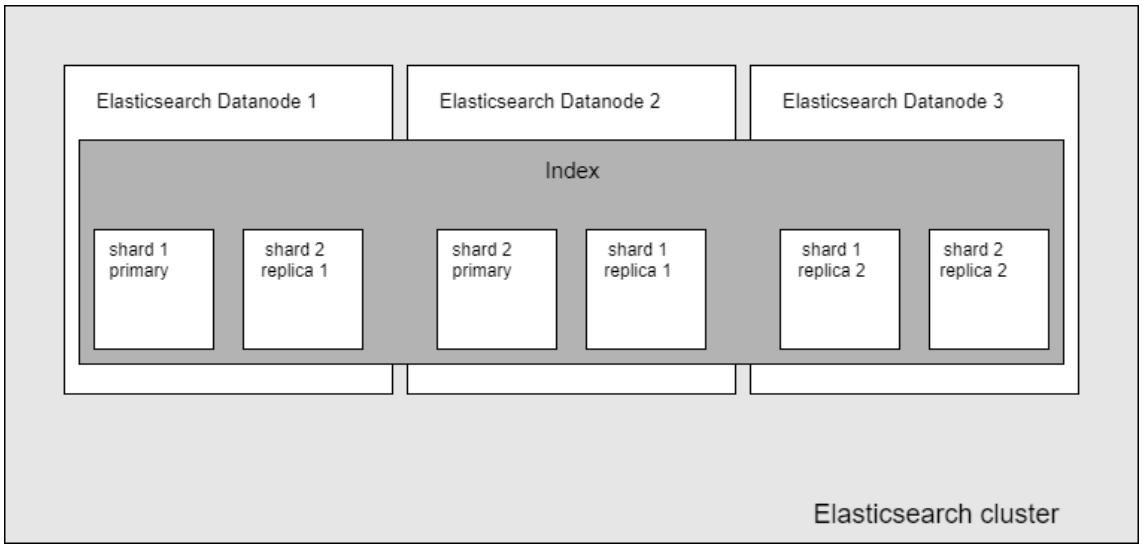
Elasticsearch käyttää Apachen Lucene kirjastoa kirjoittaakseen ja lukeakseen dataa indekseistä. Apachen Lucene kirjoittaa tiedot käänteisen indeksin muodossa, joka tarkoittaa sitä, että data järjestetään termien perusteella dokumentteihin. Tämä on päinvastainen toimintatapa verrattuna normaaliin relationaaliseen tietokantaan, joka on järjestetty dokumenttien mukaan. Käänteisessä indeksissä termipohjaiset haut ovat nopeita ja termeihin on myös lisätty lukumäärä, jolla nähtään kuinka monta kertaa tietty termi on esiintynyt dokumenteissa [12]. (Taulukko 1.)

Term	Count	Docs
4	1	<3>
Apache	1	<3>
Cookbook	1	<3>
ElasticSearch	2	<1> <2>
Mastering	1	<1>
Server	1	<1>
Solr	1	<3>

Taulukko 1. Esimerkki käänteisestä indeksistä [12].

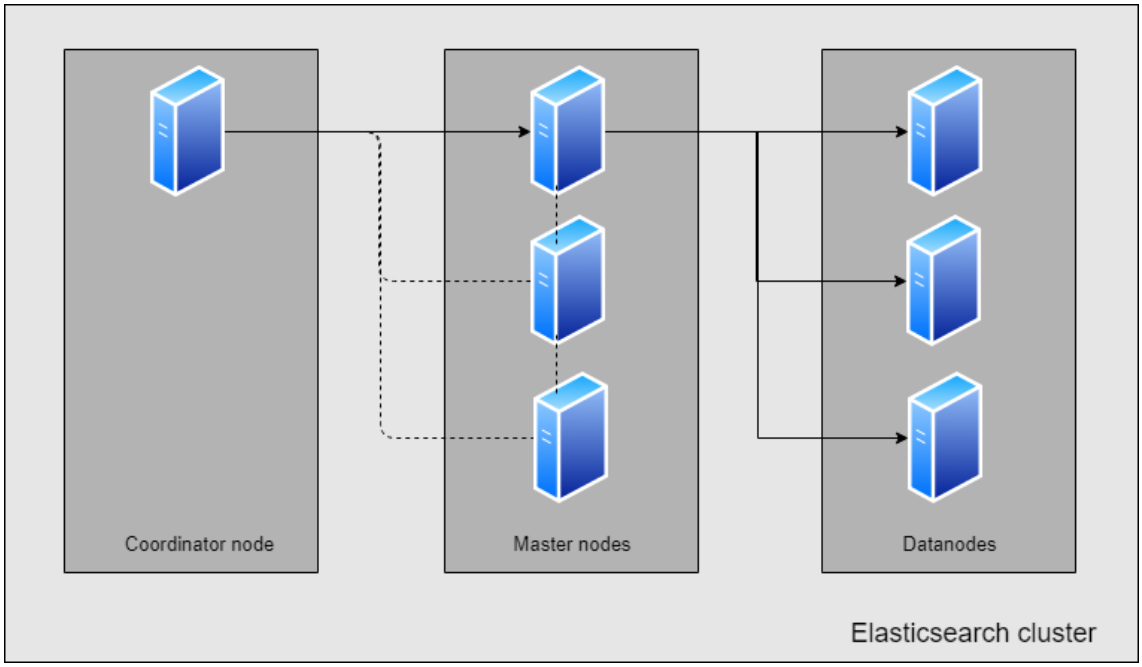
Elasticsearch jakaa dataa useisiin Lucene-indekseihin, joita kutsutaan shardeiksi eli sirpaleiksi. Indeksillä voi olla useita sirpaleita, ja niiden koko vaihtelee, mutta Elasticin mukaan maksimikoko yhdelle sirpaleelle on noin 40 Gb. Sirpaleiden koko taas riippuu indeksin koosta. Vaikka data on jakautunut indeksien osalta useisiin sirpaleisiin, näkee käyttäjä kuitenkin indeksit kokonaisina osina. Sirpaleista tehdään replikoita, eli kopioita toisille Elasticsearchin klusteriin kuuluville noodeille. Jos kyseessä on vain yhden Elasticsearchin kokonaisuus, ei replikoita voida luoda. Kopioidut replikaatit toimivat aivan kuten alkuperäiset sirpaleet, joista ne ovat otettu. Noodin, jossa alkuperäinen sirpale on,

kaatuessa Elasticsearch pystyy edelleen toimimaan normaalisti sirpaleen replikaateilla ja dataa ei menetetä. Replikaatti ei sellaisenaan pysty suorittamaan indeksöintiä, vaan noodin pitää tehdä siitä uusi primäärisirpale [12]. Kuvassa 2 nähdään indeksin jakautuminen primääri- ja replikaattisirpaleihin Elasticsearch klusterissa.



Kuva 2. Indeksin arkkitehtuuri Elasticsearch klusterissa [Tuomo Pihlasto 2019].

Elasticsearch-palvelu voidaan jakaa useampaan noodiin eli koneeseen, jolla Elasticsearch on käytössä ja se skaalautuu helposti todella suuriinkin tarpeisiin. Useasta noodista muodostuvaa Elasticsearch kokonaisuutta kutsutaan klusteriksi. Klusterin noodeja voidaan ennalta määritellä eri toimintoihin, esim. datanoodi ja mestarinoodi. Elasticsearchin toiminnan jakaminen useamman noodin klusteriksi auttaa jakamaan kuormitusta. Elasticsearch pystyy toimimaan yhdelläkin noodilla, joten klusterointi auttaa myös parantamaan palvelun saatavuutta. Elasticsearch pystyy toimimaan katkotta vaikka klusterista olisi useampi noodi alhaalla. Klusterissa on aina määriteltynä yksi mestarinoodi, joka on vastuussa klusterin tilan manageroinnista ja monitoroinnista. Mikä tahansa noodi voi olla mestarinoodi, jos klusteriin ei ole erikseen määritetty yksinomaisia mestarinoodeja, jotka eivät ole datanoodeja [12]. Kuvassa 3 on esitetty yksinkertainen Elasticsearch klusteri, jossa on roolien perusteella yksilöidyt noodit.



Kuva 3. Elasticsearch klusterin arkkitehtuuri [Tuomo Pihlasto 2019].

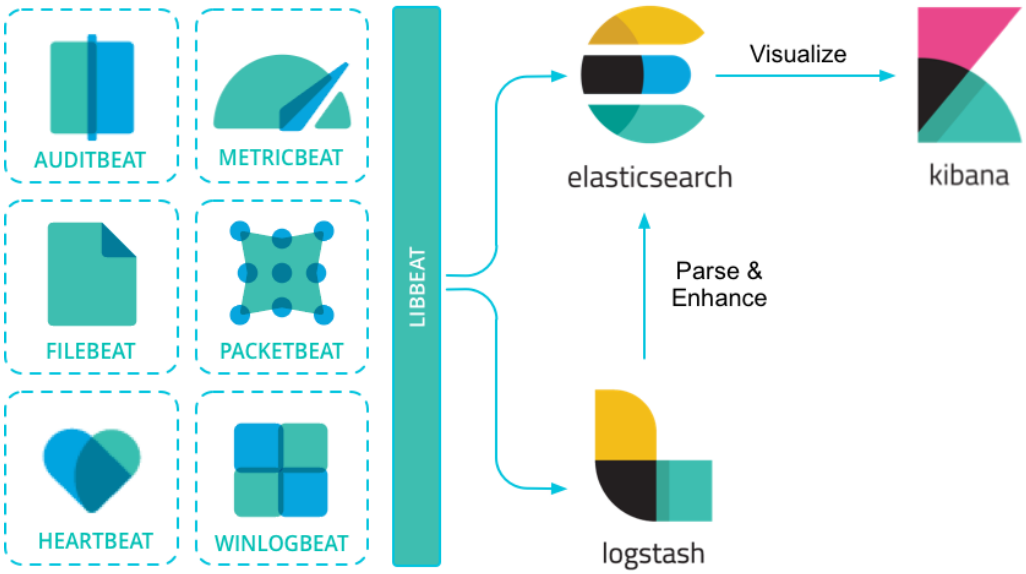
Elastic Stackin voi myös asentaa Elasticin tarjoamaan pilvipalveluun, jos ei itse halua asentaa ohjelmia palvelimilleen. Pilvipalvelun kautta on helpompi asettaa koko järjestelmä pystyyn sekä suorittaa päivityksiä. Sitä kautta myös koko klusteri on automaattisesti suojattu. Palvelulle tulee tietenkin enemmän hintaa kuin jos hoitaisi itse kaiken [8].

2.2 Beat

Beatit ovat Elastic Stackiin kuuluvia erillisiä komponentteja, jotka hakevat dataa ja lähettävät sen eteenpäin, joko suoraan Elasticsearchiin tai Logstashiin. Jokainen beat hakee palvelimelta erilaista dataa. Tässä työssä käytettävät beat-komponentit ovat Filebeat, Metricbeat sekä Heartbeat. Kaikki beat-komponentit ovat todella kevyitä palvelimien resurssien käytön kannalta, joten ne eivät aiheuta merkittävää muistin tai prosessorin käyttöä. Ne ovat kaikki avointa lähdekoodia ja niiden käyttö on ilmaista [4].

Beat-komponentit toimivat kaikki libbeat-kirjaston avulla (Kuva 4.) ja se tarjoaa myös käyttöön API:n, jolla voidaan tehdä itse rakennettuja beatteja omaan käyttöön. Libbeat-kirjasto sisältää kaikki beat-komponenttien yleiset komponentit joilla ne pystyvät lähettämään dataa eteenpäin, käsittelemään tiedostoja, lokitusta ja signaaleja. Tällä libbeat-

kirjaston tarjoamalla yleisellä rungolla, varmistetaan, että kaikki beat-komponentit ovat yhteensopivia ja toimivat yhtenäisesti [4].



Kuva 4. Dokumenttien kulku Elastic Stackissa [4].

2.2.1 Filebeat

Filebeat kerää palvelimelta eri ohjelmien lokitietoja ja lähettää ne eteenpäin keskitettyyn varastoon. Se lukee palvelimelta lokitietoja tai tiedostoja joita se on asetettu lukemaan ja lähettää kerätyt lokitiedot eteenpäin, joko suoraan Elasticsearchiin sellaisenaan tai Logstashille jäsennöitäväksi [1].

Filebeat käynnistää jokaiselle konfiguroinnista asetetusta polusta löytämälleen lokitiedostolle kerääjän, joka tarkkailee ja lähettää halutut lokiviestit eteenpäin. Tämä kerääjä lukee yhtä tiedostoa rivi kerrallaan ja lähettää jokaisen uuden tapahtuman tiedot libbeatille. Asetuksia pitää muuttaa jos tarkkailtava ohjelma voi antaa lokitapahtumia, jotka menevät useammalle riville. Muuten Filebeat tunnistaa jokaisen rivin uudeksi lokitapahtumaksi. Libbeat kerää lokitapahtumat ja lähettää kootut lokitiedot Filebeatille määriteltyyn ulostuloon eli Elasticsearchiin tai Logstashiin, riippuen konfiguraatioasetuksista [2].

Filebeat muistaa mikä lokitapahtuma on lähetetty viimeksi ja jos se ei saisi yhteyttä sille asetettuun ulostuloon, ei yhteyskatkoksen aikana tulleet lokitapahtumat jää lähettämättä, sillä heti yhteyden palautuessa Filebeat jatkaa toimintaansa siitä kohtaa mihin se jäi. Jokaiselle sisääntulossa asetetulle lokitiedostolle on oma uniikki tunniste, jolla Filebeat tunnistaa lokitiedoston vaikka sen nimeä olisi muutettu tai tiedostoa siirretty. Näin Filebeat varmistaa, että jokainen lokitapahtuma lähetetään vähintään kerran. Jos Filebeat suljetaan samalla kun sen prosessi on lähettämässä lokitapahtumaa eteenpäin, Filebeat ei vastaanota ulostulolta vastausta, että lokitapahtuma on vastaanotettu. Tässä tapauksessa on mahdollista että Filebeatin käynnistyessä se lähettää kyseisen lokitapahtuman uudestaan, jolloin syntyy kaksi samanlaista lokitapahtumaa. Tämä voidaan estää asettamalla Filebeat odottamaan ennen ohjelman sulkemista [2].

2.2.2 Metricbeat

Metricbeat kerää eri moduulien avulla asetetuin väliajoin järjestelmästä ja eri ohjelmista metriikkaa ja statistiikkaa, kuten palvelimen levyn tallennustilan ja prosessorin käyttöä. Metricbeattiin on saatavilla useita eri moduuleja, jotka keräävät eri ohjelmien tietoja. Esimerkkinä system-moduuli, joka kerää palvelimelta sen suorituskykyyn liittyviä tietoja. System-moduuli on aktivoitu oletuksena Metricbeatin asennuksen jälkeen. Tietojen keräämisen jälkeen Metricbeat lähettää tiedot sille merkittyyn ulostuloon, joko suoraan Elasticsearchiin varastoitavaksi tai Logstashiin jäsennettäväksi [5].

2.2.3 Heartbeat

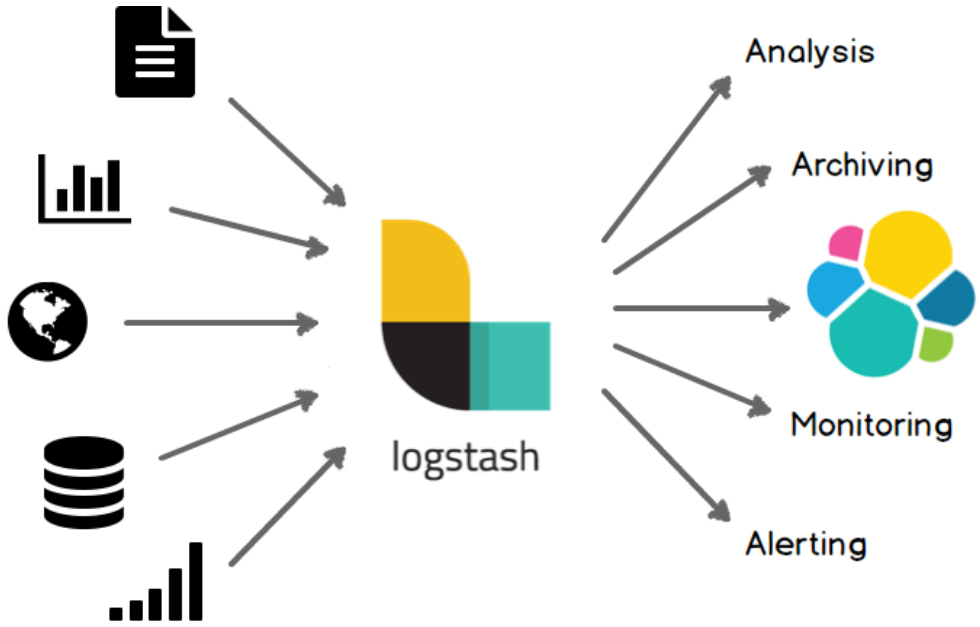
Heartbeat tarkistaa palvelimilta eri web-pohjaisten palveluiden toiminnan kutsumalla aktiivisesti osoitteita, joissa kyseiset palvelut toimivat. Tällä hetkellä Heartbeat tukee vain ICMP, TCP ja HTTP -tyyppisiä kutsuja. TCP- ja HTTP-monitoroinnit tukevat myös SSL/TLS -teknologioita ja joitakin salaukseen liittyviä proxyjä [3]. Metricbeatista poiketen, Heartbeat kertoo tarkemmin palvelimen tilasta, kun halutaan tietää, onko jokin yksittäinen palvelu saatavilla palvelimella.

Yhteen Heartbeat-asennukseen voidaan määrittää useita tarkasteltavia monitoreita, kuten sen konfigurointitiedostoissa mainitaan. Monitori tarkoittaa Heartbeatin

tapauksessa osoitetta ja sen yhtä tai useampaa porttia, joissa valvottava palvelu toimii. Nämä monitorit voidaan määrittää, joko suoraan Heartbeatin pääkonfiguraatioon tai niille voidaan tehdä oma erillinen kansio ja asettaa konfiguraatio lukemaan sitä. Jälkimmäinen ratkaisu on parempi suuremmassa mittakaavassa, kun tarkkailtavia monitoreita on useampia. Heartbeatin konfiguraatioon voidaan määrittää tarkentavasti, mitä vastausta se odottaa monitorin osoitteesta. Oletuksena Heartbeat hyväksyy vain 200-ok, vastauksen palvelimelta joka tarkoittaa että kyseisessä portissa oleva palvelu on käynnissä ja vastaa Heartbeatin kutsuun. Tässä työssä käytetään Heartbeatin 6.3.0 versiota, joka on vielä betaversio ja tässä versiossa ei ole mahdollista käyttää monitoroiden konfigurointien säilytykseen erillistä kansiota. Monitoreiden tarkkailua voidaan aikatauluttaa monipuolisesti asetuksissa, esimerkiksi yksi monitori voidaan asettaa tarkkailemaan yhden palvelun toimintaa 07.00–18.00 ja toinen jatkuvasti 10 s:n välein [3].

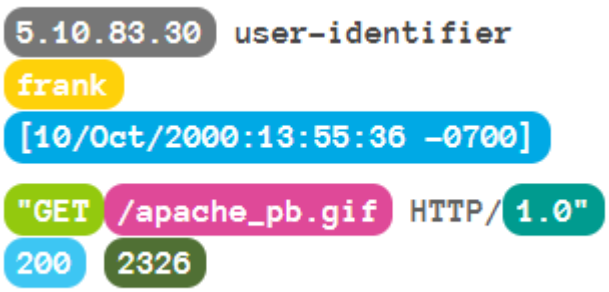
2.3 Logstash

Logstash pystyy vastaanottamaan tietoja useista eri lähteistä, kuten beateilta tai toiselta Logstashilta samanaikaisesti ja lähettää tiedot eteenpäin Elasticsearchiin tai johonkin muuhun tietokantaan varastoitavaksi (Kuva 5.). Logstash tukee useita eri lähteitä valmiilla lisäosilla ja tarjoaa myös mahdollisuuden tehdä omia lisäosia tarpeisiinsa sopivaksi. Logstashiin voidaan asettaa useita kanavia (engl. pipeline), joilla on jokaisella oma konfiguraationsa. Kanavia käytetään, kun kyseisellä palvelimella on useita lähteitä, joista Logstash vastaanottaa tietoa ja jokaisesta lähteestä usein tulee hyvin erilaista dataa. Kanaviin on silloin selkeämpi tehdä konfiguraatiomuutoksia sekä kanavan läpi kulkevat tiedot voidaan lähettää eri Elasticsearchin indekseihin [6].



Kuva 5. Logstash [6].

Yksi Logstashin hyöty on sen kyky suodattaa kaikesta sille syötetyistä tapahtumista yksittäiset tiedot omiksi osiksiin (Kuva 6.). Se käyttää tähän grok-kielellä kirjoitettuja regex-pohjaisia suodattimia. Grok-suodatin perustuu siihen, että lokiviesteistä voidaan olettaa niiden noudattavan aina samaa kaavaa, eli jokainen lokiviesti rakentuu samoista osista tietyssä järjestyksessä [7]. Tätä toistuvaa mallia noudattaen voidaan rakentaa grok-suodatin, joka tietää, että lokiviesti alkaa aikaleimalla, jonka jälkeen tulee lokitustaso jne. Nämä toistuvat elementit lokiviesteissä voidaan siten nimetä omiin kenttiinsä, joilla niitä voidaan hakea.



Kuva 6. Grokilla suodatetut ja poimitut kentät dokumentista [9].

2.4 Kibana

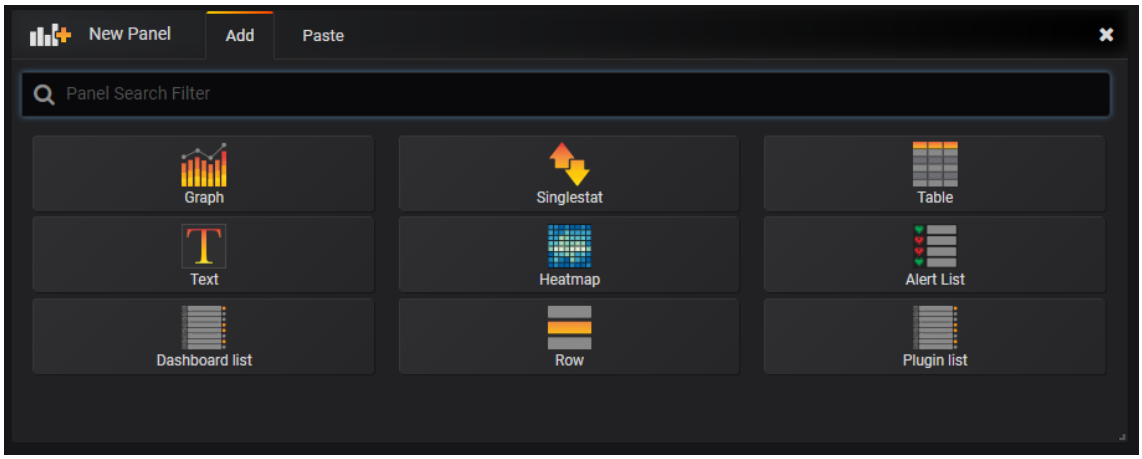
Elastic Stackin viimeinen osa, joka näyttää sen todelliset hyödyt, kun kaikki kerätty ja pilkottu data tuodaan esiin graafeina, kuvaajina ja taulukoina. Kibanan kautta on myös mahdollista asettaa hälytyksiä (vain maksullisissa versioissa) kerätylle datalle. Kibana on kuin kehitys- ja hallintapaneeli Elasticsearchille. Kibanassa pystytään tarkastelemaan Elasticsearchin eri indeksejä ja niissä olevaa dataa. Dataa voidaan analysoida erittäin tarkasti, jokainen yksittäinen tapahtuma on pilkottavissa ja tapahtuman jokaista kenttää ja sen arvoja voidaan tarkastella.

Kibanassa pystytään myös ylläpitotasolla tarkastelemaan Elasticsearchin indeksejä, kuten vaikka eri kentille asetettuja arvoja tai tietoja itse indeksistä. Kibanan avulla voidaan tarkastella myös koko klusterin tilaa ja siihen kuuluvien yksittäisten noodien toimintaa ja toimintahistoriaa. Kibanan käyttöliittymän kautta on mahdollista hallinnoida Elasticsearchin indeksejä, kuten poistaa vanhoja ja tehdä uusia indeksejä sekä myös päivittää ja tehdä uusia templaatteja indekseille.

2.5 Grafana

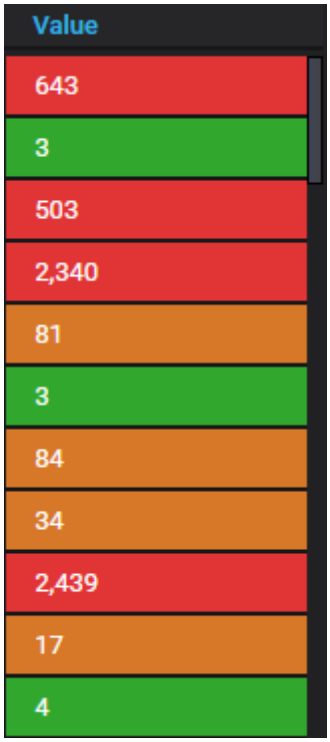
Grafana ei kuulu Elastic Stackiin ja on kilpaileva tuote sen omalle vastikkeelle, Kibanalle. Ohjelmana Grafana on samantyylinen kuin Kibana, mutta AgentIT käyttää Grafanaa tiedon visualisoimiseen ja hälytysten tekoon Kibanan sijaan, sillä Grafanassa on parempi käyttöliittymä ja tiedon visualisointi on helpompaa. Käyttäjien luonti ja pääsynhallinta on Grafanassa toteutettu helppokäyttöisesti käyttöliittymän kautta ja tässä on valtava ero Kibanan ilmaisversioon, jossa ei ole mahdollista asettaa roolipohjaista pääsynhallintaa (muuttuu versiosta 6.8 alkaen). Grafanassa on kolme eri tason käyttäjäroolia, jotka ovat admin, editor ja viewer. Tason admin käyttäjällä on ylläpitäjän oikeudet kyseisessä ympäristössä. Editor-tason käyttäjällä on oikeudet luoda uusia paneeleita ja muokata jo olemassa olevia paneeleita. Viewer käyttäjäroolilla on vain oikeus katsella jo tehtyjä paneeleita [10]. Grafanassa on olemassa vielä niinkutsuttu super-admin, jolla on automaattisesti kaikki oikeudet jokaiseen kyseiseen Grafanaan tehtyyn ympäristöön ja organisaatioon. Tämä super-admin luodaan automaattisesti Grafanan asentuessa ja se on oletuskäyttäjä, jolla kirjaudutaan käyttöliittymään ensimmäistä kertaa.

Grafanassa on mahdollista määritellä datalähteeksi Elasticsearchin eri indeksejä, ja näistä indekseistä voidaan hakea Lucene- pohjaisella haulla haluttuja kenttiä, sanomia ja tapahtumia. Grafana tukee useita eri tietokantoja datalähteinä. Tietoa voidaan visualisoida monilla erilaisilla paneeleilla (Kuva 7.). Tieto visualisoidaan paneeleihin valitsemalla paneelille tietolähde, jonka jälkeen kirjoitetaan haku, jolla haluttu tieto saadaan esiin [10].



Kuva 7. Grafanan (v5.4.1) paneelivaihtoehdot.

Asennuksessa tulevien peruspaneelien lisäksi Grafana tarjoaa sivuillaan useita erilaisia ladattavia paneeleita [11]. Lisäpaneelit on ladattava Grafanaan liitännäisen avulla ja asennuksen jälkeen Grafana on käynnistettävä uudelleen, jotta uudet paneelit ovat valittavissa. Lisäpaneelien lisäksi Grafanassa pystyy jakamaan itse tekemiään paneelinäkymiään muiden ladattavaksi ja testattavaksi Grafanan omilla sivuilla. Grafanan snapshot-ominaisuutta käyttämällä pystyy jakamaan sen hetkisen näkymän ja liittämään näkymässä olevat tiedot siihen, jolloin vastaanottaja saa linkin interaktiiviseen näkymään siitä hetkestä ja niillä tiedoilla, jotka tilannekuvan ottohetkellä näkyvät. Paneeleita pystyy muokkaamaan vapaasti esimerkiksi lisäämällä värejä tiettyihin arvoihin tai muuttaa kuvaajan yksiköitä, akseleita ja graafia [11]. Taulukoissa voidaan näyttää minkä tahansa kentän saamia arvoja, jotka löytyvät dokumentista. Kuvassa 8 on käytetty värikoodausta paneelin arvoissa.

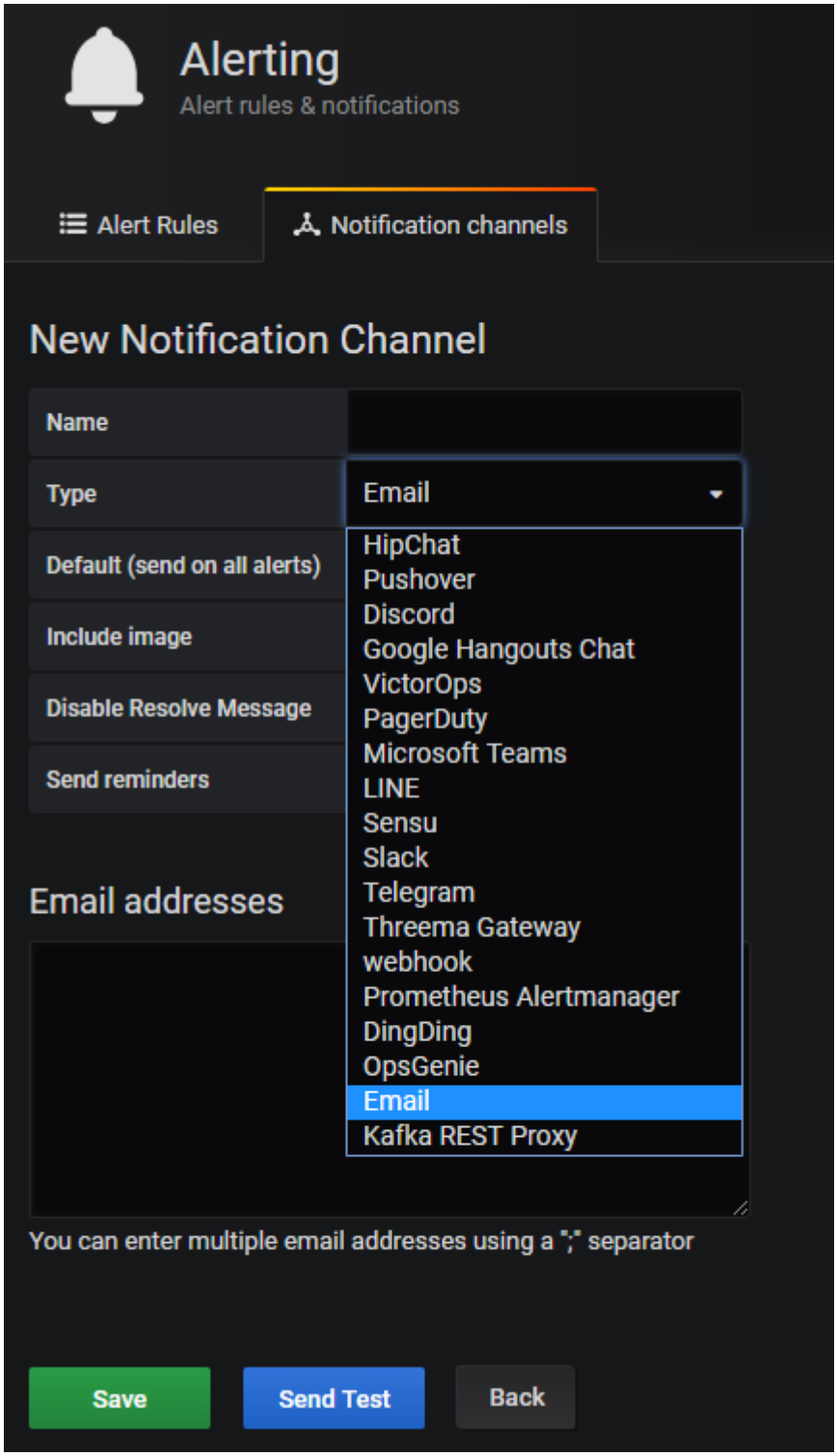


Kuva 8. Värikoodaus taulukon arvoissa.

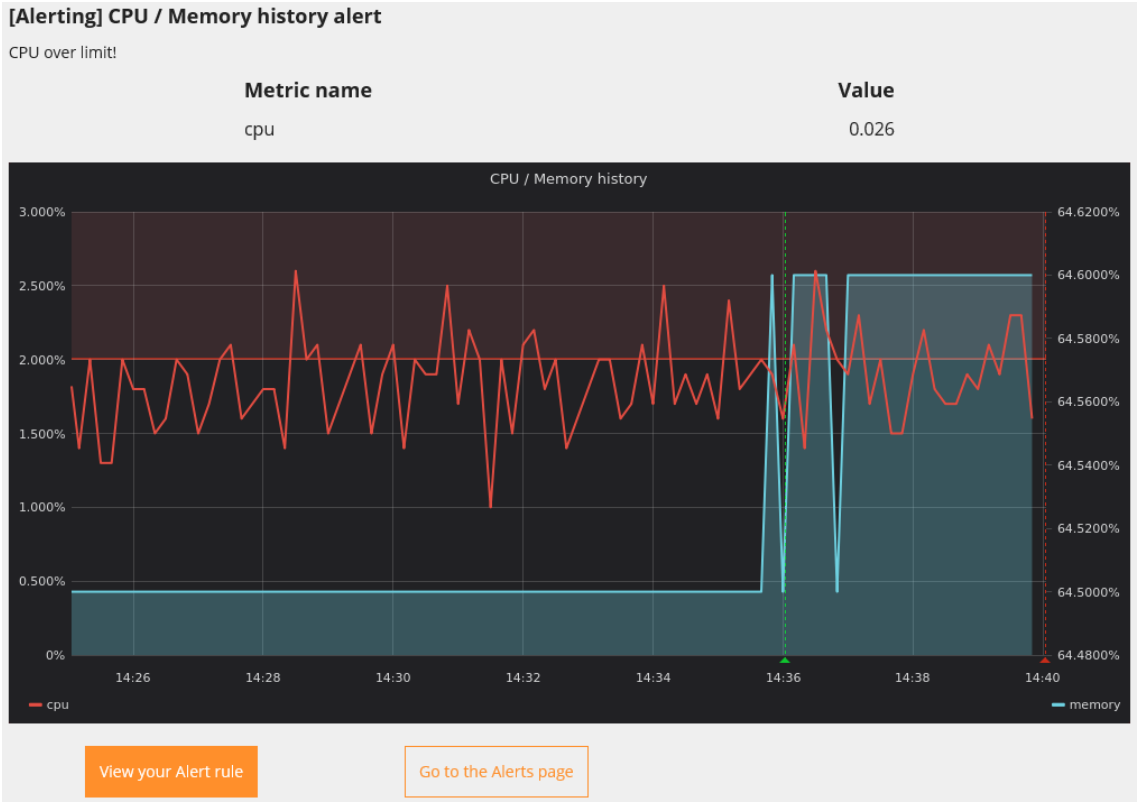
Grafanan hälytyksillä on tiettyjä tiloja, joihin ne voivat mennä. Nämä tilat ovat Ok, Pending, Alerting, Paused ja No Data. Ok-tila on silloin päällä, kun hälytyksen arvot eivät riko hälytykseen asetettuja arvoja. Hälytyksen rajojen rikkoutuessa se siirtyy alerting-tilaan. Hälytykset voidaan ottaa väliaikaisesti pois käytöstä, jolloin ne muuttuvat paused-tilaan. Kun paused-tilassa oleva hälytys aktivoidaan taas, muuttuu sen tila pending-tilaan, jolloin hälytys määrittelee ylittävätkö sen saamat arvot hälytyksen rajat. Hälytys voi joutua pending-tilaan myös, jos sille on asetettu hälytyksen asetuksissa for-kenttään jokin arvo. For-kenttä hälytyksessä tarkoittaa sitä, että hälytys voi hetkellisesti mennä yli asetettujen rajojen, mutta jos se jatkuu kauemmin kuin on for-kentässä annettu arvoksi, siirtyy se alerting-tilaan. Jos hälytys ei saa erikseen haettua dataa tai haetun datan arvoa ei voida määrittää (null), muuttuu hälytys no data -tilaan. Grafana säilyttää myös historiatietoja menneistä hälytyksen tilojen muutoksista sekä kaikkia hälytyksiä tarkastellessa, jokaisesta näkyy erikseen, miten kauan hälytys on ollut nykyisessä tilassa [10].

Grafanaan pystyy määrittelemään useita eri notifikaatiokanavia (Kuva 9.), joiden kautta tieto hälytyksistä menee eteenpäin. Kanavalle on mahdollista asettaa erilaisia tyyppejä, joista yleisin on email eli sähköpostiviesti. Hälytyksen voi myös asettaa lähettämään muistutuksia tietyin väliajoin, jos se ei palaa ok-tilaan. Include image -asetus liittää

hälytyksen viestiin kuvan kyseisestä hälytyksen kuvaajasta, jossa näkyy hieman datan historiaa ennen hälytyksen laukeamista. Kuvassa 10 vihreä katkoviiva osoittaa hetken, jolloin hälytyksen tila on muuttunut ok-tilaan ja punainen katkoviiva, kun hälytys on mennyt alerting-tilaan. Viesti kertoo kuvassa, mikä arvo on ylittänyt hälytysrajan ja myös sen arvon sillä hetkellä. Viestiin sisältyy myös linkit, joilla pääsee suoraan hälytyksen asetuksiin sekä hälytyssivulle, jossa näkyy kaikki määritellyt hälytykset sekä niiden tämänhetkiset tilat [10].



Kuva 9. Grafanan hälytysten notifikaatiokanavien määrittely.



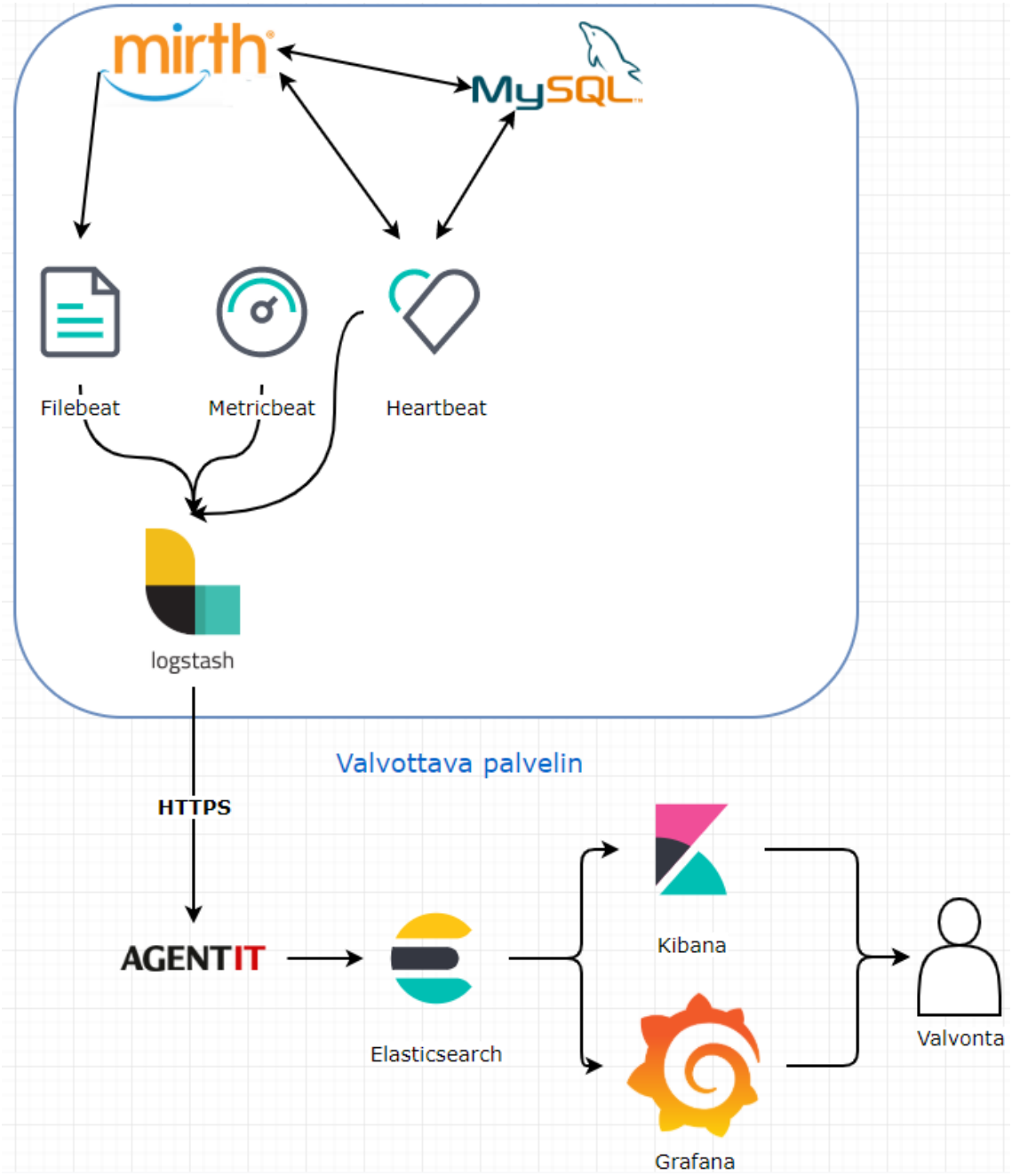
Kuva 10. Esimerkki Grafanan sähköpostihälytyksen viestistä.

3 ELASTIC STACK -TOTEUTUS

Työn tavoitteena oli nopeuttaa NextGen Connect -integraatio-ohjelmiston virheiden ja poikkeamien valvontaa tekemällä Elastic Stackia hyödyntäen ratkaisu, jolla luetaan integraatio-ohjelmiston lokia. Luetut lokiviestit siirretään ratkaisun avulla AgentIT:n jo käyttämään Elasticsearchiin, jonka kautta ne voidaan visualisoida Grafanaan. Jokaisesta asennetusta Elastic Stackin komponentista piti poistaa oletuksena oleva Elasticsearchin ulostulo pois käytöstä ja sen tilalle aktivoida Logstashin ulostulo, sillä Logstashin avulla halutaan rikastaa ja jäsennellä tietoa hyödyllisempään muotoon.

Elastic Stackin komponentit asennettiin linux-palvelimelle käyttämällä Elasticin omaa repositoria eli varastoa, josta linux-palvelimet voivat hakea asennustiedostoja. Komponenttien toiminnan varmistamiseksi hyödynnettiin Kibanaa, tarkastelemalla tuleeko uudesta asennuksesta dataa klusteriin. Jos dataa ei huomattu Kibanassa, lähdettiin tutkimaan komponenttien lokitietoja ongelman selvittämiseksi. Valvontaratkaisua asennettaessa palvelimelle, on palvelimelle jo asennettuna ja toiminnassa NextGen Connect -integraatio-ohjelma. Asiakkaan palvelimelle, oli se sitten AgentIT:n tarjoama tai asiakkaan oma, ei asenneta Elasticsearchia, ellei asiakas niin halua. Valvontakomponentin osana toimii AgentIT:n oma Elasticsearch klusteri, joka vastaanottaa asiakkaalle asennettavien komponenttien keräämän datan. Tällöin käytetään myös AgentIT:n omaa Grafanaa datan visualisoimiseen ja hälytysten tekemiseen. Kuva 11 näyttää valvontakomponentin rakenteen ja miten se liittyy AgentIT:n ympäristöön. Valvontakomponenttiin voidaan tietenkin lisätä muita valvottavia kohteita palvelimelta, eikä ne tule häiritsemään NextGen Connect integraatio-ohjelmiston valvontaa millään tavalla. Grafanaan voidaan tehdä asiakkaalle oma ympäristö (organisaatio), jossa on näkymät vain heidän palvelimilta kerätystä datasta.

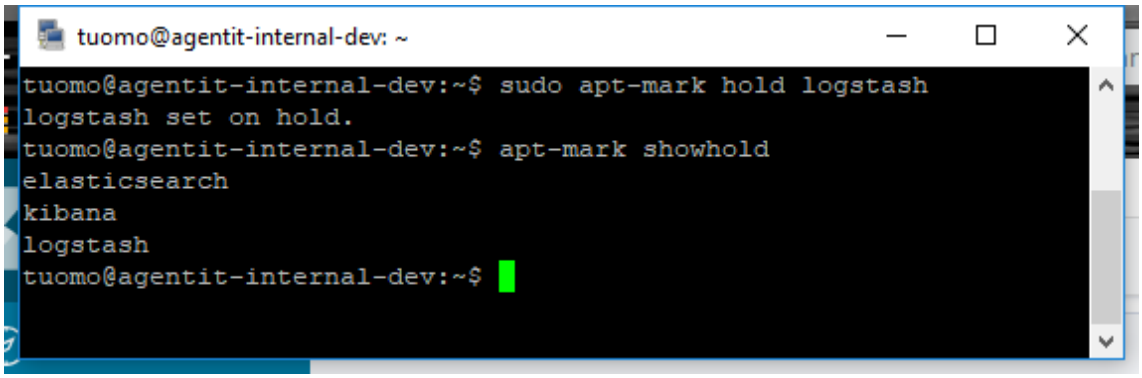
Asennusprosessista konfiguraatioineen tehtiin AgentIT:n dokumentointijärjestelmään dokumentit. Jokaiselle asennetulle komponentille tehtyyn dokumentointiin on kirjattu askel askeleelta asennusprosessin vaiheet ja konfiguraatioiden muutokset. Dokumentit on eritelty komponenttien osalta eri dokumenteiksi.



Kuva 11. Valvontakomponentti osana AgentIT:n kokonaisuutta [Tuomo Pihlasto 2019].

Valvontakomponentin kokonaisuudesta tehtiin ensin testiympäristöön toimiva konsepti, joka kehittämisen ja testaamisen jälkeen tuotiin ensimmäiselle asiakkaalle tuotantokäyttöön. Aihe oli minulle uusi ja siksi asian opetteluun kului aluksi paljon aikaa. Ikioma testiympäristö auttoi paljon, kun oli mahdollisuus tehdä lähes kaikkea ilman vaikutusta tuotantoympäristöön. Komponenteista asennettiin 6.3.0 versio, joka on myös tämänhetkisen AgentIT:llä käytössä oleva Elastic Stackin versio. Elasticin komponenttien versio saa

olla vain niin suuri kuin käytössä olevan Elasticsearchin versio, sillä Elasticsearch ei tue itseään uudempien komponenttien versioita. Tulevaisuudessa komponentteja tullaan päivittämään ja se pitää ottaa huomioon myös konfiguraatioissa ja erityisesti niiden dokumentoinnissa. Komponenttien asennukset tehtiin Linux-käyttöjärjestelmälle, mutta konfiguraatiot ovat täysin samat myös Windows-palvelimille. Jokainen komponentti asetettiin myös palvelimen käynnistyksen yhteydessä käynnistyväksi sekä komponenttien versiot lukittiin, jotteivat ne päivity automaattisesti. Kuvassa 12 komponentin versio lukitaan jonka jälkeen listataan lukitut komponentit.



Kuva 12. Komponentin version lukitseminen ja lukituksen tarkistaminen konsolilla.

Työssä käytettäviä työkaluja olivat grok, kitty ja notepad++. Grok-työkaluna toimi verkkosivu, jossa pystyi rakentamaan grok-suodattimia käyttäen itse antamiaan lokiviestejä testinä. Ilman työkalua grok-suodattimen teko olisi todella haastavaa, sillä sitä ei pystyisi testaamaan ilman sen käyttöönottamista Logstashiin ja silloinkin suodattimen saamat lokiviestit eivät olisi tarpeeksi monimuotoisia todistaakseen suodattimen monipuolisuuden erilaisten lokiviestien suhteen. Verkkosivun tarjoaman työkalun kanssa sen sijaan tuloksen pystyi näkemään heti. Työkalussa oli myös mahdollisuus syöttää käsin mahdollisimman erilaisia esimerkkilokiviestejä, joiden avulla rakennetusta suodattimesta saataisiin mahdollisimman kattava. Kitty on työkalu, jolla voi ottaa omalta tietokoneeltaan konsoliyhteyden toiseen linux-palvelimeen ja tehdä siellä asennuksia ja konfiguraatioita. Kitty:a hyödynnetään aina asiakkaiden palvelimilla työskenteltäessä, sillä fyysisesti asiakkaiden palvemille meno ei ole tarpeellista. Notepad++ on todella yleinen apuväline tiedostojen kanssa työskenteltäessä. Sillä voidaan tarkastella ja muokata kaikenlaisia eri tiedostoja niiden tiedostomuodosta riippumatta.

3.1 Filebeat

Asennuksessa Filebeat asetetaan lähettämään paikalliseen Logstashiin dataa, jonka kautta data lähetetään AgentIT:n klusteriin. Filebeat asetetaan lukemaan NextGen Connectin lokitiedostoja, joten konfiguroinnissa polku laitetaan osoittamaan sinne mihin ohjelma kirjoittaa lokia ja asetetaan Filebeat lukemaan aina sen hetkistä lokitiedostoa. NextGen Connect -integraatio-ohjelman lokeissa on ns. juokseva luku. Sen hetkinen lokitiedosto, johon myös kirjoitetaan, on ilman järjestyslukua, mutta kyseisen lokitiedoston tullessa täyteen, asetetaan täyttyneeseen lokitiedostoon juokseva luku ja uusi lokitiedosto ilman lukua luodaan. Koska NextGen Connect -integraatio-ohjelmiston tiedetään lokittavan jokaiseen lokiviestiin sen tason voidaan Filebeatin konfiguraatiotiedostoon asetetaan sen poimivan vain rivit, joilla mainitaan ’ERROR’, ’WARN’ tai ’FATAL’. Näiden lokitustasojen viestit ovat niitä, jotka kiinnostavat valvontaa. Kaikki integraatio-ohjelmiston viestit eivät mene lokitiedostossa yhdelle riville, joten Filebeatin konfiguraatiotiedostoon pitää lisätä malli, jolla kaikki uudet lokiviestit alkavat. Lokiviestien tarkastelun perusteella, jokainen uusi lokiviesti alkaa aina lokitustasolla jonka jälkeen tulee lokiviestin aikaleima. Tämän perusteella konfiguraatiotiedostoon asetetaan regexillä tehty malli, jolla jokainen uusi lokiviesti tulee alkaa. Jo luetut tiedostot pysyvät tallessa Elasticsearchin indekseissä niille määritellyn ajan. Filebeatin konfiguraatitiedostoon asetetaan lisäkenttiä, joilla yksilöidään, että mitkä dokumentit tulevat juuri tältä palvelimelta ja juuri tästä lähteestä. Näiden kenttien avulla pystytään myöhemmin Grafanassa asettamaan yksilöllisiä kuvaajia sekä Kibanan kautta tarkastella tuleeko juuri kyseisestä palvelimesta ja lähteestä dokumentteja Elasticsearchiin asti.

Samalla periaatteella Filebeat konfiguroidaan lukemaan palvelimen järjestelmälokia. Uudemmissa versioissa Filebeat saa erillisen system-moduulin, jolla saadaan lokiviestit suoraan palvelimen jäjestelmälokilta jäsennettynä ilman erillistä grok-suodattimen tekoa Logstashiin. Palvelimen järjestelmälokia ei ennen tarkasteltu, mutta nyt sitä halutaan tarkastella, sillä sinne tulee esimerkiksi Elasticin komponenttien virheviestejä sekä muita viestejä, jotka kiinnostavat valvontaa. Näistä kerätyistä viesteistä ei kuitenkaan tule hälytyksiä, mutta ne ovat ongelmanratkontatilanteissa avuksi. Viesteille tehdään Grafanaan oma näkymä, mutta ne näkyvät tietysti myös Kibanassa, kuten kaikki Elasticsearchiin tallennetut viestit.

3.2 Metricbeat

Asennuksessa Metricbeat asetetaan lähettämään paikalliseen Logstashiin dataa, jonka kautta data lähetetään AgentIT:n Elasticsearch klusteriin. Metricbeatin asennuksessa automaattisesti käyttöönotettu system-moduulin konfiguraatioon tehtiin pieni muutos, jotta kyseiseltä palvelimelta saataisiin vastaavaan tahtiin samat metriikat, kuten jo muilta AgentIT:n valvonnan alla olevilta palvelimilta. Asennuksessa annettiin konfiguraatiotiedoston kautta lisäkenttiä, joita käytetään dokumenttien yksilöimiseen, jolloin voidaan tarkasti tietää mistä palvelimelta ja miltä ohjelmalta dokumentti on peräisin. Yksilöity tieto auttaa myöhemmin kun tiettyjä dokumentteja pitää etsiä Elasticsearchin indekseistä.

System-moduulin konfiguraatiota piti muuttaa hieman oletusarvoista. Joka kymmenes sekunti palvelimelta otetaan esim prosessorin sekä muistin käyttö ja molemmista otetaan viisi eniten käyttävää prosessia ja ne järjestetään suurimmasta pienimpään. Koko palvelimen käynnissäoloaikaa otetaan ylös 15 minuutin välein ja levyjen tilaa tarkkaillaan minuutin välein.

3.3 Heartbeat

NextGen Connect -integraatio-ohjelmiston valvonnan kannalta on tärkeää tarkkailla porttia, jossa itse ohjelma toimii sekä porttia, jossa sen käyttämä MySQL tietokanta toimii. Heartbeat on konfiguroitu monitoroimaan palvelimella olevan NextGen Connect -integraatio-ohjelman lokaalia osoitetta ja porttia, jossa se toimii. Integraatio-ohjelman lisäksi tarkkaillaan sen käyttämää MySQL-tietokantaa kuuntelemalla porttia, jossa se lokaalisti toimii. Kummallakin näistä monitoreista on omat yksilölliset konfiguraatiot. Konfiguraatiotiedostoon lisätään monitoreiden kohdalle eri kenttiä, kuten name, tags yms. Nämä kentät siirtyvät dokumenttien tullessa monitoroidusta osoitteesta Elasticsearchiin talteen, josta niitä voidaan hakea ja analysoida. Grafanan näkymissä hyödynnetään näitä kenttiä, kun halutaan yksilöidä, mistä monitorista on kyse. Asennettu versio on vielä beta-versio, joten päivittäessä pitää erityisesti ottaa huomioon, jos jotain käytettyä ominaisuutta on muutettu tai käytetyille ratkaisuille löytyy parempi vaihtoehto.

3.4 Logstash

Logstash kuuntelee omasta sisääntulostaan beat-komponenttien viestejä ja odottaa sieltä aktiivisesti dokumentteja, jotka menevät määritellysti omille kanavilleen käsiteltäviksi, jos niitä on erikseen määritelty. Kanavissa Logstash ohjaa dokumentit grok-suodattimen läpi (Kuva 13.), joka jäsentää lokituksen eri kenttiin ja lisää dokumenttiin arvon, jonka perusteella Elasticsearch tietää mihin indeksiin dokumentti säilötään. Tehdyllä NextGen Connect -grok-suodattimella lokiviestistä eroitetaan lokiviestin taso (loglevel), aikaleima (timestamp), lokiviestin aiheuttaneen kanavan nimi ja id (identification) sekä itse viesti. Vaikka lokiviesti jaetaan osiin, alkuperäinen viesti kokonaisuudessaan säilyy silti. Lopuksi Logstash lähettää jäsennetyt kentät Elasticsearchiin varastoitavaksi eri indekseihin. Logstashin kautta suodatettavat viestit ohjataan AgentIT:lla eri Elasticsearchin indekseihin, riippuen siitä mistä viestit ovat peräisin. Metricbeat, Filebeat ja Heartbeat saavat jokainen oman indeksin joka vaihtuu aina määritellyin väliajoin uuteen. Vanhat indeksit jäävät talteen Elasticsearchiin määritellyksi ajaksi ja niistä voidaan edelleen etsiä ja analysoida tietoa.

```
if "Filebeat" in [tags] and "Mirth" in [tags] {
 grok {
  match => {
   "message" => "^(?m)%{LOGLEVEL:loglevel}\s{1,2}%{TIMESTAMP_ISO8601:timestamp}\s\[%{DATA:from}\]\s(?<shortmessage>.{1,256})%{GREEDYDATA}$"
  }
 }
```

Kuva 13. Logstashin konfiguraatiossa oleva grok-suodatin.

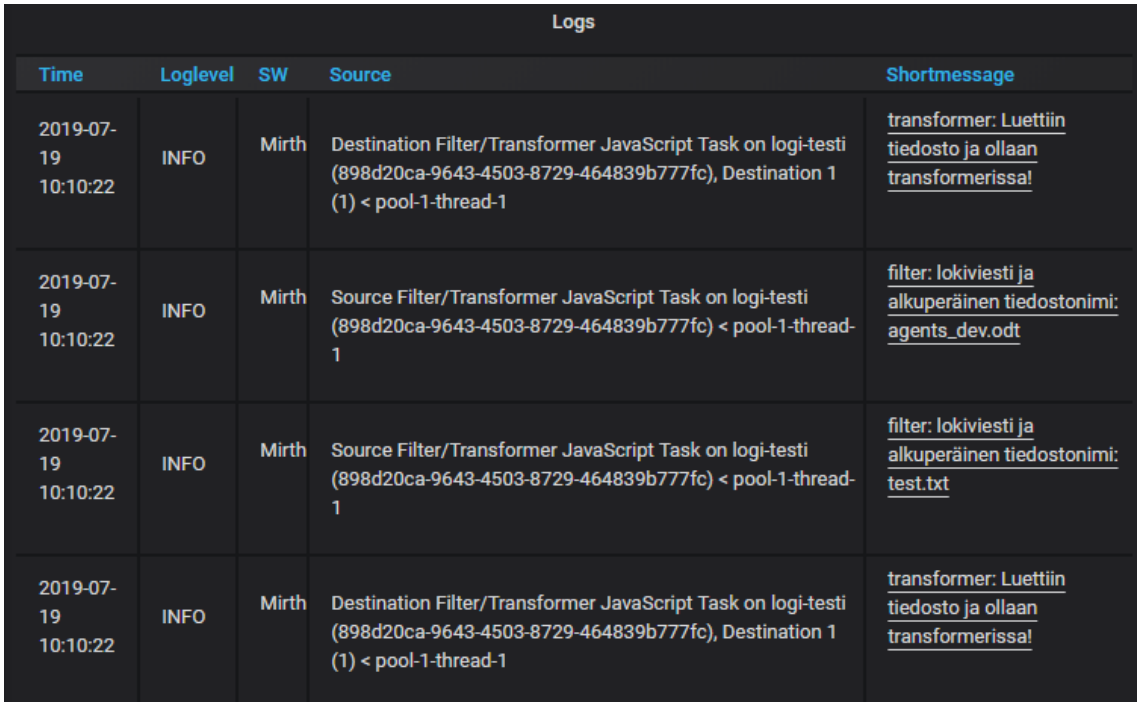
Logstashiin asennettiin JMX-input-lisäosa, jolla voidaan hakea ja lähettää NextGen Connect -integraatio-ohjelmasta tämän käyttämiä JVM-tietoja. Tällä kerätyllä tiedolla saamme tietää ohjelman käyttämistä resursseista. JMX-input käyttää konfiguraationaan erillistä JSON-tiedostoa, johon on määritelty mitä kenttiä ohjelman JVM:stä halutaan seurata. Mahdollisia valvonnalle hyödyllisiä kenttiä tarkastellessa päädyin valitsemaan kentät, jotka kertovat ohjelman muistin käytön ja prosessorin käytön sekä ohjelman käynnissäoloajan. Logstashiin piti tehdä konfiguraatio, jotta JMX-input tietäisi mitä tietoja ohjelman JVM:stä otetaan. Konfiguraatio tehtiin JSON-kielellä ja se asetettiin Logstashin konfiguraatiokansioon. Logstashiin tehtiin uusi kanava, joka ohjattiin käyttämään tätä JMX-inputin konfiguraatiota. Kanavan sisääntuloksi asetettiin JMX-input, jolloin sinne tulisi vain ohjelman JVM tiedot, kun muut palvelimelta kerätyt tiedot menisivät toisen kanavan kautta.

NextGen Connect -integraatio-ohjelman JVM aktivoitiin lähettämään dataa Logstashin JMX-input-lisäosaan. JVM:sta otetaan integraatio-ohjelman käytettävissä oleva muisti, prosessorin käyttö ja käynnissäoloaika. Jotta NextGen Connect -integraatio-ohjelmasta saataisiin JVM:n tietoja, pitää sieltä avata JMX etätarkkailu omaan porttiin. Porttiin yhteydenotto vaatii tunnuksen ja salasanan, jotka ovat määritelty omiin tiedostoihin ja asetettu integraatio-ohjelmiston asennuslokaation alle omaan kansioon. SSL-suojaus otettiin pois käytöstä, jolloin porttiin tarvitaan vain erillinen käyttäjä ja salasana. Logstashin kirjautuessa integraatio-ohjelmiston JMX etätarkkailuun se käyttää JMX-konfiguraatiotiedostoon asetettuja tietoja.

3.5 Grafana

Grafanaan määritetään Elasticsearch tietolähteeksi ja siten Grafanalla on pääsy lukemaan Elasticsearchin indeksejä. Grafanassa Elasticsearchin tietolähteet erotetaan indeksien nimillä erikseen, eli Metricbeat, Filebeat ja Heartbeat ovat jokainen omana tietolähteenä. Tietolähde etsii kaikki Elasticsearchin indeksit kyseiseen tietolähteeseen, jotka sopivat sille merkittyyn indeksin kaavaan, esim voidaan määritellä [heartbeat]-* jolloin tietolähteeseen sisältyy kaikki indeksit, jotka alkavat heartbeat-. *-merkki hyväksyy minkä tahansa lopun indeksille, jolloin on mahdollista, että aikaleimat, eikä versionumerot vaikuta, vaan tietolähteeseen saadaan kaikkien heartbeat-indeksien tiedot Elasticsearchista.

Näkymät rakennettu NextGen Connect -integraatio-ohjelman lokeille, jotka järjestetään ensisijaisesti aikaleiman mukaan. Näkymän taulukossa näytetään lokituksen taso, ohjelma, mistä lokitus on peräisin sekä itse lokiviestin sisältö 256 merkkiin asti. Kaikki lokiviestit eivät ole niin lyhyitä, että ne mahtuisivat maksimissaan 256 merkkiä sisältävään kuvassa 14 näkyvään shortmessage-kenttään. Lyhennettyyn viestiin on upotettu linkki, jonka valitsemalla aukeaa lokiviesti kokonaisuudessaan sellaisena kuin se on poimittu lokitiedostosta, jolloin nähdään myös se osa, mikä ei mahtunut tuon 256 merkin sisään.

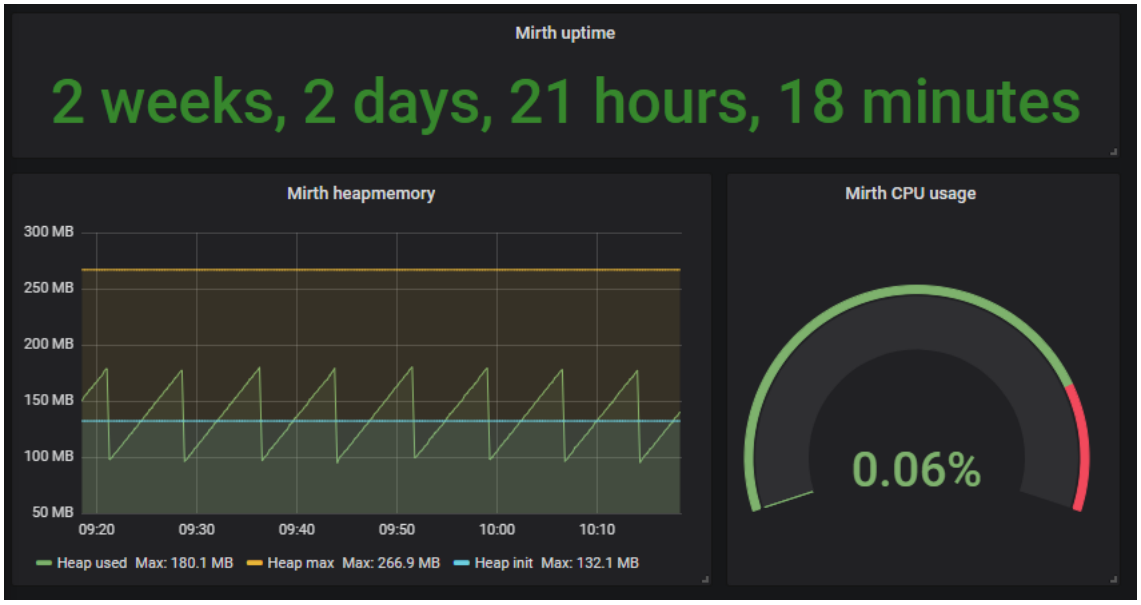


Logs

Time	Loglevel	SW	Source	Shortmessage
2019-07-19 10:10:22	INFO	Mirth	Destination Filter/Transformer JavaScript Task on logi-testi (898d20ca-9643-4503-8729-464839b777fc), Destination 1 (1) < pool-1-thread-1	transformer: Luettiin tiedosto ja ollaan transformerissa!
2019-07-19 10:10:22	INFO	Mirth	Source Filter/Transformer JavaScript Task on logi-testi (898d20ca-9643-4503-8729-464839b777fc) < pool-1-thread-1	filter: lokiviesti ja alkuperäinen tiedostonimi: agents_dev.odt
2019-07-19 10:10:22	INFO	Mirth	Source Filter/Transformer JavaScript Task on logi-testi (898d20ca-9643-4503-8729-464839b777fc) < pool-1-thread-1	filter: lokiviesti ja alkuperäinen tiedostonimi: test.txt
2019-07-19 10:10:22	INFO	Mirth	Destination Filter/Transformer JavaScript Task on logi-testi (898d20ca-9643-4503-8729-464839b777fc), Destination 1 (1) < pool-1-thread-1	transformer: Luettiin tiedosto ja ollaan transformerissa!

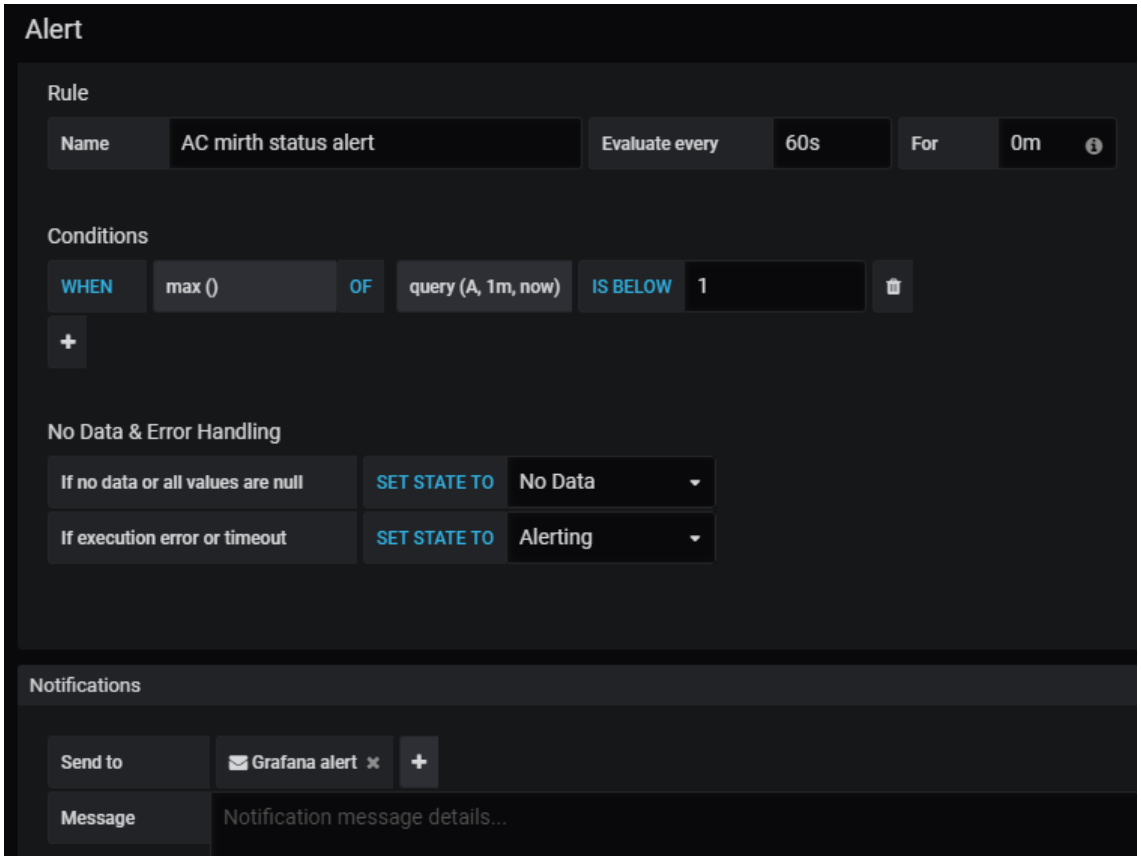
Kuva 14. Grafanaan tehty NextGen Connect (Mirth) lokien infopaneeli.

JMX-monitoroinnin avulla saadut JVM-tiedot on esitetty kuvassa 15 testikoneella tehtyyn Grafanan näkymään. Siinä näkyy integraatio-ohjelmiston käynnissäoloaika, sen käytössä oleva ja tällä hetkellä kuluttama muisti sekä prosessorinkäyttö. Käynnissäoloaikaan on myös asetettu värikoodaus, jolloin tietyn rajan ylittyessä fontin väri muuttuu. Jos palvelimia on useita samassa näkymässä, niin on helpompi nähdä värikoodien perusteella, jos jonkin palvelimen integraatio-ohjelmisto on hiljattain käynnistynyt uudelleen.



Kuva 15. Grafanaan tehny JMX-näkymä.

Hälytykset Grafanassa pitää tehdä kuvaajaan, joka aiheuttaa tietyn tyylisten hälytysten tekemiseen pieniä haasteita. Heartbeat-hälytysten tapauksessa hälytys sidottiin monitor.status-kenttään, joka voi saada arvokseen joko up tai down. Nämä arvot annetaan sillä perusteella saako Heartbeat onnistuneesti yhdeyden monitoroitavaan osoitteeseen vai ei. Hälytys laukeaa, jos minuutin aikana ei tule yhtään dokumenttia, jossa monitor.status-kentän arvo on up. Kuvassa 16 hälytyksen edellytykset tarkastetaan 60 sekunnin välein, eli jos A haun nykyhetkestä edellisellä minuutilla saatu maksimiarvo on alle 1, hälytys laukeaa ja lähettää sähköpostihälytyksen erikseen määritellyn notifikaatiokanavan kautta.



Kuva 16. Hälytyksen asetukset.

Tietyn kanavan lokitusta pystytään poimimaan, kunhan saadaan tarpeeksi yksilöivä asia eroteltua viestistä, jota ei muusta kanavasta tule, ettei vahingossa suodateta muita tärkeitä viestejä pois näkymästä. Tämä hoituu helpoiten, kun asetetaan kanava lokittamaan jokin tietty viesti esim. ”kanava X suoritti toimenpiteen” voidaan silloin Filebeat poimimaan juuri tämän viestin sisältävät lokiviestit Filebeatin include_lines-toiminnolla. Testivaiheesta tuotantoon siirtyessä tuli joistakin testivaiheen kanavista turhia hälytyksiä, kun kehitystyötä tehtiin edelleen, joka korjattiin lisäämällä Grafanan hälytyksen hakuun ehto ”NOT from:*TEST*”, joka ei hae kuvaajaan niitä dokumentteja, joissa kenttä from sisältää kirjainyhdistelmän TEST. Tämä kenttä asetetaan NextGen Connect -integraatio-ohjelmiston kanavan nimeen lokiviestien jäsentämisessä.

4 HYÖDYT

4.1 Yritys

Tekemästäni valvontanäkymästä saadaan AgentIT:lle valmis kokonaisuus, joka voidaan ottaa käyttöön jokaiselle palvelimelle, jossa NextGen Connect -integraatio-ohjelma on käytössä. Malliasennuksesta on tehty kattava dokumentointi, jonka avulla valvontakomponentit voidaan asentaa ja konfiguraatioita muuttaa, jotta komponenteista saadaan uutta dataa. Asiakaskohtaiset eroavaisuudet ja hälytykset pitää kuitenkin ottaa huomioon, mutta niihinkin löytyy esimerkkejä jo tehdyistä ratkaisuista. Ratkaisu helpoittaa ohjelmaan liittyvää valvontaa löytämään nopeammin virheen syyn. Virheisiin pystytään reagoimaan nopeammin ja niiden aiheuttaja pystytään selvittämään helposti ja siten myös toiminta virheen ratkomiseksi voi alkaa ripeämmin.

Hyvin konfiguroituna valvonnan ei tarvitse keskittyä palvelimien aktiiviseen valvomiseen, sillä komponenteilla toteutetun valvonnan avulla virheiden sattuessa tai asetettujen rajaarvojen ylittyessä tapahtumasta lähtee hälytys, joka kertoo mikä aiheutti hälytyksen. Valvonnan ja hälytysten konfigurointi on yksilöllistä jokaiselle asiakkaalle ja niille tehdään muutoksia, jotta löydetään oikeat arvot sekä saadaan poistettua mahdolliset väärät hälytykset. Valvonnan laadun parantuessa asiakkaat ovat tyytyväisiä ja yritys saa uusia referenssejä. Uudet referenssit autttavat saamaan lisää näkyvyyttä yritykselle ja sitä kautta uusia asiakkaita.

4.2 Asiakas

Asiakas säästää tällä ratkaisulla aikaa, kun ei tarvitse itse erikoistaa henkilöä valvomaan ohjelmaa. Asiakas voi myös päättää, ettei halua valvontaa AgentIT:ltä jolloin asiakkaan oma valvontahenkilö pystyy suuresti hyötymään tehdystä valvontanäkymästä. Suurimpia hyötyjä on ajansäästö, sillä valvontakomponentti tarkkailee aktiivisesti palveluita, jolloin työntekijöillä jää enemmän aikaa muuhun. Virheiden sattuessa hälytyksien avulla tieto saadaan nopeasti eteenpäin, jolloin korjaustoimenpiteet voidaan aloittaa lyhyemmässä ajassa. Nopean reagoinnin johdosta mahdolliset palvelukatkokset tai muut virheet saadaan korjattua normaalia nopeammin, minimoiden häiriöistä aiheutuvat kulut ja haitat.

5 LOPUKSI

Tämän opinnäytetyön tavoitteena oli soveltaa Elastic Stackia NextGen Connect -integraatio-ohjelmiston valvomiseen. Toteuttu kokonaisuus tuli olla helposti toistettavissa useammille palvelimille hyödyntäen asennuksesta tehtyjä dokumentteja. Kokonaisuuden tuli yhdistyä AgentIT:n jo käyttämään Elastic Stackiin.

Elastic Stack oli aiheena täysin uusi, joten työn alussa meni suuri osa ajasta aiheen opiskeluun ja ymmärtämiseen. Aiheen ymmärtämisessä auttoi Elasticin oma nettisivu, josta löytyy kattavasti useita eri ohjeita sekä kaikki mahdollinen dokumentaatio. Aiheen sisäistämisessä oli ensisijaisen tärkeää oma testiympäristö, jossa harjoitella eri ominaisuuksia. Myös AgentIT:n oman ympäristön tutkiminen perusymmärryksen muodostuessa auttoi merkittävästi. Työssä käytetty Grafana oli myös täysin uusi ohjelma, joka oli kuitenkin kokonaisuutena pienempi ja siten nopeampi ja helpompi sisäistää kuin Elastic Stackin kokonaisuus.

Tehty dokumentointi antaa ohjeet vain yleiseen asennukseen, jolla saadaan kaikki lokiviestit napattua Elasticsearchiin. Ratkaisu on todella joustava, ja sitä voidaan muokata eri tarkoitusten mukaan, eikä kannata tyytyä jo toimivaan ratkaisuun, vaan etsiä muutoksia, jotka ovat parempia kyseiseen käyttötarkoitukseen. Komponenttien konfiguraatiotiedostot on usein kirjoitettu YAML-muotoon, joka on hyvin tarkka siinä käytettävän syntaksin kanssa. Esimerkkinä yksi välilyönti liikaa tai liian vähän rivin alussa ja komponentti ei käynnisty virheen takia.

Grafanaan tehtäessä asiakkaille omia organisaatioitaan on otettava huomioon, että asiakkailla on mahdollisuus nähdä myös muiden asiakkaiden dataa, jos he menevät tekemään omia näkymiään. Grafanaan asetetut datalähteet hakevat dataa Elasticsearchin indekseistä, joihin ei ole erikseen eritelty asiakkaita vaan kaikkien asiakkaiden samantyylinen data menee samaan indeksiin. Jotta asiakas pystyisi näkemään toisen asiakkaan dataa, pitäisi hänen pystyä tietämään, mitä he etsivät, ja tuntea myös Grafanan käyttöliittymä. Ensimmäisessä organisaatiossa, joka AgentIT:n Grafanaan tehtiin, tämä ratkaistiin asettamalla asiakkaiden rooliksi Grafanassa viewer, jolla ei ole oikeuksia luoda tai muuttaa näkymiä eikä hakuja, joilla muiden tietoja olisi mahdollista löytää.

LÄHTEET

[1] Elastic, 2019. Filebeat overview. Viitattu 12.4.2019. https://www.elastic.co/guide/en/beats/filebeat/7.0/filebeat-overview.html

[2] Elastic, 2019. How Filebeat works, What is a harverter?, How does Filebeat keep the state of files?, How does Filebeat ensure at-least-once delivery?. Viitattu 12.4.2019. https://www.elastic.co/guide/en/beats/filebeat/7.0/how-filebeat-works.html

[3] Elastic, 2019. Heartbeat overview. Viitattu 12.4.2019. https://www.elastic.co/guide/en/beats/heartbeat/7.0/heartbeat-overview.html

[4] Elastic, 2019. Overview. Viitattu 12.4.2019. https://www.elastic.co/guide/en/beats/libbeat/7.0/beats-reference.html

[5] Elastic, 2019. Metricbeat overview. Viitattu 12.4.2019. https://www.elastic.co/guide/en/beats/metricbeat/7.0/metricbeat-overview.html

[6] Elastic, 2019. Logstash Introduction, The Power of Logstash, Logs and Metrics, Data Stores and Streams, Easily Enrich Everything, Choose Your Stash. Viitattu 2.5.2019. https://www.elastic.co/guide/en/logstash/current/introduction.html

[7] Elastic, 2019. Grok Basics. Viitattu 29.7.2019. https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html

[8] Elastic, 2019. Viitattu 12.4.2019. https://www.elastic.co/products/elastic-stack

[9] Elastic, 2019. Inputs, Filters, Outputs, Extensibility. Viitattu 2.5.2019. https://www.elastic.co/products/logstash

[10] Grafana, 2019. Introduction, Rule Config, For, Conditions, Notifications, Alerts State History & Annotations. Viitattu 19.7.2019. https://grafana.com/docs/alerting/rules/

[11] Grafana, 2019. Data Source, Organization, User, Panel, Query Editor, Dashboard. Viitattu 19.7.2019. https://grafana.com/docs/guides/basic_concepts/

[12] Kuc, Rafal, and Marek Rogozinski, 2013. Mastering ElasticSearch. Viitattu 15.4.2019. http://ebookcentral.proquest.com/lib/turkuamk-ebooks/detail.action?docID=1362591.

[13] NextGen, 2019. Viitattu 22.7.2019. https://www.nextgen.com/products-and-services/integration-engine