Effektiv logghantering och monitorering av IT-infrastruktur med Elastic stack

Martin Salonen

Examensarbete

Informationsteknik

2018

Martin Salonen

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informationsteknik
Identifikationsnummer:	6625
Författare:	Martin Salonen
Arbetets namn:	Effektiv logghantering och monitorering av IT-infrastruktur med Elastic stack
Handledare (Arcada):	Magnus Westerlund
Uppdragsgivare:	Svenska litteratursällskapet i Finland r.f.

Sammandrag:
Effektiv hantering, monitorering och felsökning är utmaningar i den moderna IT-miljön. Proaktivt arbete är att föredra istället för reaktivt. Detta hjälper även till med att bättre förstå kugghjulen i infrastrukturen och därmed minimera oförväntade problem och avbrott i systemen. Under uppdrag av arbetsgivaren Svenska litteratursällskapet i Finland r.f. (SLS) är målet med detta arbete att implementera ett centraliserat system för logghantering och monitorering av IT-infrastrukturen. Målet med detta system är att hjälpa IT-teamet få en bättre överblick över infrastrukturen och effektivera upptäckande av problem och snabba upp problemlösningsprocessen. Systemet som skall implementeras består av programserien Elastic stack. Denna serie består främst av tre delar: Elasticsearch för indexering av data, Logstash/Beats för inmatning och behandling av data och Kibana för grafiskt användargränssnitt och visualisering av data. Utöver detta skall också expansionspaketet X-Pack implementeras, vilket av Alerting-modulen kommer att användas för alarmering under definierade omständigheter. Arbetet är uppdelat i fyra delar. I den första delen beskrivs infrastrukturen hos uppdragsgivaren och utmaningar i modern systemadministration definieras. Proaktivt och reaktivt arbete inom systemadministration behandlas också. I den andra delen presenteras Elastic stack och dess komponenter, vad de är och vad deras roll är i systemet. I den tredje delen planeras och implementeras systemet. Ett kluster för systemet planeras och val av specifikationer och komponenter görs med motiveringar. Sedan implementeras systemet. Elasticsearch installeras och konfigureras på tre olika servrar. Inmatning och behandling av data från diverse källor sätts upp med hjälp av Logstash och Beats. Kibana installeras med konfiguration och användbara visualiseringar som del av skräddarsydda instrumentbräden sätts upp. Alarmering implementeras också med Alerting modulen från X-Pack. Till implementering av systemet hör också underhåll av Elasticsearch med hjälp av programvaran Curator. I den sista delen av arbetet utvärderas det uppsatta systemet och en sammanfattning görs. Feedback tas in av IT-teamet, där systemets användbarhet utvärderas med tanke på användarvänlighet och nytta i systemadministration. Till sist tas en syn på framtida utvidgad användning av systemet.

Nyckelord:	Elastic stack, logghantering, systemadministration, visualisering, statistik, SLS
Sidantal:	41
Språk:	Svenska
Datum för godkännande:	5.6.2018

DEGREE THESIS	
Arcada	
Degree Programme:	Information technology
Identification number:	6625
Author:	Martin Salonen
Title:	Efficient log management and monitoring of IT-infrastructure using Elastic stack
Supervisor (Arcada):	Magnus Westerlund
Commissioned by:	Society of Swedish Literature in Finland

Abstract:
Efficient administration, monitoring and troubleshooting is a challenge in the modern IT setting. Proactive action is preferable to reactive, while aiming for a better understanding of the underlying components in the IT infrastructure, thereby minimizing downtime in services. The purpose of this thesis is to set up a centralized log management and monitoring system for the IT infrastructure of the client, Swedish society of literature in Finland. This system will be built on the Elastic stack, an open source software suite. The suite consists mainly of three parts; Elasticsearch that indexes data, Logstash/Beats for ingestion of data and Kibana for graphic interface and visualization. This system's purpose is to help the IT staff in making it faster and easier to identify and troubleshoot problems. An alerting function will also be implemented, where the system will alert the IT staff by email under specified circumstances. The thesis consists of four parts. First, the IT infrastructure is described and what challenges there are in modern system administration. In the second part, Elastic stack and its components are examined. The third part consists of the actual implementation of Elastic stack, data is ingested into Elasticsearch and visualizations created in Kibana, as well as alerting. The last part of the thesis consists of evaluation of the new system, including user-friendliness and actual usefulness in system administration. This is done via receiving feedback from the IT team. Lastly, a look into the future where further implementation of the system is evaluated.

Keywords:	Elastic stack, logging, system administration, visualization, statistics, SLS
Number of pages:	41
Language:	Swedish
Date of acceptance:	5.6.2018

INNEHÅLL

Figurer

Tabeller

1 INTRODUKTION

Modern systemadministration kan vara krävande. Datorisering och automatisering är dagens melodi då det gäller de flesta funktioner i ett företag. E-post, telefoni, filsystem, videokonferens, personaladministration, säkerhetskopiering och säkerhetsnycklar är endast fåtal exempel på system som faller under en IT-avdelning. Det räcker inte heller med upprätthållande av system, utan också utveckling av det. Utöver detta hör det oftast också till IT att ge stöd åt sina användare, ofta med kort varsel (Lindström 2018).

I och med dessa krav, har utvecklingen inom IT skett otroligt snabbt under senaste år. Molnteknologier erbjuder automation och tillgänglighet. Big Data är ett koncept som blir mera eftertraktat av företag för att förbättra deras konkurrenskraft. Med Big Data menar man förenklat ”stora datamängder”. Insamling och sparande av all möjlig information och konsolidering av det, presentation och tolkning ses som en väldigt värdefull tillgång (Dragland 2013).

En typisk IT infrastruktur består ofta av hundratals olika tjänster. Stora framsteg då det gäller virtuella tjänster har gjort det möjligt att sprida ut tjänster över många servrar, som samtidigt erbjuder mycket redundans. Då man tidigare hade en server med flera tjänster på en fysisk maskin, så kan man istället ha multipla virtuella servers på en fysisk maskin med diverse tjänster som tidigare var på en server, på flera olika. Detta kan anses vara bra på flera sätt, men det ökar också komplexiteten i infrastrukturen märkvärt (Carpenter 2014).

Utöver detta har man otaliga rörliga delar i nätverket, såsom switchar, routers och brandmurar. Exemplen nämnda i början kan också läggas till. Hur ska man kunna följa med och administrera dessa effektivt utan att endast dessa uppgifter blir ett heltidsjobb? Ett viktigt hjälpmedel till detta är centraliserad logghantering, monitorering och alarmering (Cordray 2015).

1.1 Syfte

Syftet med detta arbete är att identifiera utmaningar och problem med systemadministration, samt motivera och demonstrera hur ett centraliserat logghanteringssystem kan vara ett kraftfullt verktyg i detta. Istället för att behöva logga in på diverse servrar för att titta på loggfiler, så ska man istället kunna effektiv titta och göra sökningar i loggfilerna på ett

ställe. Utöver detta ska det också implementeras visualisering med data från dessa loggfiler, där man kan retroaktivt och i realtid se trender i data. Alarmering ska också implementeras, där IT-teamet kan under bestämda omständigheter alarmeras då något har gått fel. Med hjälp av dessa verktyg ska systemadministrationsarbetet effektiveras genom att ge bättre översikt över infrastrukturen och bättre kunna lösa problem då de inträffar.

1.2 Avgränsning

Arbetet består av både teori och praktik. Teorin beskrivs främst som bakgrund och motivering för praktiska genomförandet. Praktiska implementationen avgränsas också i hur många system skall kopplas till logghanteringssystemet. Som ”proof-of-concept” (conceptvalidering) kommer behandling av olika sorters loggfiler från diverse källor att implementeras. Detaljerad installation av komponenterna kommer inte att beskrivas i sin helhet, dock kommer i bilagorna läggas alla kommandon och några konfigurationsfiler som använts.

2 SYSTEMADMINISTRATION

Systemadministration går ut på att erbjuda tjänster, som används i en organisation för att utföra arbete. Till systemadministration hör därmed utveckling, implementering och upprätthållande av dessa tjänster (Limoncelli 2017 s. 283).

2.1 Utmaningar i systemadministration

Datorteknologi för hårdvara och programvara utvecklas i snabb takt. I och med detta har också kraven på relaterade tjänster ökat. Till detta hör snabbhet i implementation och minimal mängd avbrott i tjänsten.
Teknologier såsom virtuella maskiner gör det lätt att snabbt sätta upp nya tjänster. Detta medför att mängden tjänster kan snabbt öka. Som exempel så består uppdragsgivarens infrastruktur som kort översikt av följande komponenter:

- Runt 100 virtuella servrar

- Runt ett dussin Cisco switchar
- Några dussin trådlösa nätverksstationer
- Flera olika datalagringssystem, inklusive backup system
- Flera avbrottsfria kraftförsörjningssystem

Med en bemanning på tre personer i IT-teamet så är det en hel del att administrera. Användarstöd hör också till uppgifterna.

2.2 Reaktiv och proaktiv problemlösning

Det finns i princip två olika sätt att arbeta med systemadministration, reaktivt och proaktivt. Reaktivt är som själva ordet indikerar, ett reagerande arbete, ett problem uppstår någonstans och då behöver man lösa det. Beroende på hur allvarligt problemet är så kan man vara tvungen att sluta med allt man håller på med just då och istället ta itu med problemet. Detta kunde vara t.ex. allvarligt hårdvarufel, virusattack, o.s.v. Med tur så behöver man inte direkt reagera, utan istället kan man vid ett senare tillfälle undersöka problemet, t.ex. användarproblem och milda prestandaproblem. Hoppeligen kan man efter att problemet blivit löst kunna utforska vad som orsakade det, och då också se till att det inte händer igen (More 2016 s. 54-55).

Proaktiv problemlösning syftar till att man arbetar med att lösa problem förrän de händer. Detta går mycket ut på aktiv planering och monitorering. Detta kan t.ex. vara regelbunden kontroll på diverse system och se till att de fungerar som de ska, t.ex. skivutrymme på en server är tillräcklig och avsaknad av oroväckande felmeddelanden i loggboken. Att göra detta manuellt kan vara dock väldigt tidskrävande, speciellt då det gäller hundratals komponenter. En lösning på detta är automatiserande av monitorering, t.ex. ett skript som kontrollerar en server och skickar epost ifall någonting är fel. En bättre lösning är dock centraliserad monitorering och logghantering (Limoncelli 2017 s. 673 ff). Att ha insyn i infrastrukturen via ett ställe sparar tid och friger tid till att arbeta med annat. Via visualisering av data är det enkelt att se trender och vid problemfall kan man vid behov söka upp korrelation med andra data och möjligtvis hitta problemorsakande källor. Ett exempel på detta kan vara att en webbserver tjänst som slutar fungera med jämna mellanrum. Vid detta fall kan man ta en titt på processoranvändningen och antal besökare. Om man ser

en spik i dessa precis innan tjänsten slutar fungera, så kan man dra slutsatsen att prestandan inte räcker till. Då kan man lösa problemet t.ex. genom att lägga till en processor till servern för att hjälpa till med belastningen.

2.3 Relevanta egenskaper att mäta

I en IT-infrastruktur kan det finnas otaliga mängder av egenskaper att mäta. Det kan vara till mycket nytta att definiera de viktigaste måtten. Dessa varierar beroende på tjänsten. För en webbtjänst kan tekniska relevanta egenskaper vara responstid och hur länge det tog att ladda sidor (Palmer 2002 s. 155-158). Det kan därefter vara viktigt att mäta resursanvändning för servern, t.ex. processoranvändning, minnesanvändning och lagringsutrymme. Detta underlättar möjlig felsökning där man kan hitta korrelation och kausalitet med olika mått. T.ex., om responstiden på en webbtjänst plötsligt blir hög, kan man hitta korrelation i hög processoranvändning. Kausaliteten kan sedan bestämmas ifall responstiden förbättras ifall man ger servern en till processorkärna att arbeta med.

3 ELASTIC STACK

Elastic Stack är benämningen på kombination av tre/fyra programvaror, Elasticsearch, Logstash/Beats och Kibana. Tillsammans skapar de ett kraftfullt system för analys av stora datamängder (Big Data). Denna kombination var tidigare också känd som ELK Stack (**E**lasticsearch, **L**ogstash, **K**ibana). Alla tre programvaror är gratis och baserar sig på öppen källkod, dock skapas största delen av utvecklingen av företaget Elasticsearch BV. Företaget säljer också instanser av Elastic Stack som SaaS-molntjänst (Software as a Service). I ett nötskal erbjuder Elastic Stack ett centralt system för att samla ihop data och metrik från otaliga tjänster, servrar och apparater. I detta system kan man sedan behandla dessa data, presentera det visuellt och automatisera åtgärder på basis av det.
Elastic Stack består alltså av minst tre komponenter (se tab.1).

Tabell 1. Elastic stack komponenter

Kibana	Användargränssnittet, visualisering av data
Elasticseach	Förvaring, indexering och analys av data
Logstash/Beats	Behandling och inmatning av data in till Elasticsearch

Utöver detta så finns det avgiftsbelagda komponenter i expansionspaketet X-Pack. Den erbjuder expanderade funktioner till Elastic Stack, såsom extra säkerhet, alarm, monitorering, med mera.

Figur 1. Grafisk illustration av Elastic Stack.

3.1 Elasticsearch

Elasticsearch är i sin grund en sökmotor, som är baserad på Lucene informationssökningssystemet. Lucene har en lång historia (första utgivning 1999) och är populär i instanser där man behöver kunna indexera och söka i hel text (McCandless et al. 2010). Den är populär som sökmotor på nätet, t.ex. Twitter använder det för sin realtidssökning (MG Siegler 2010).

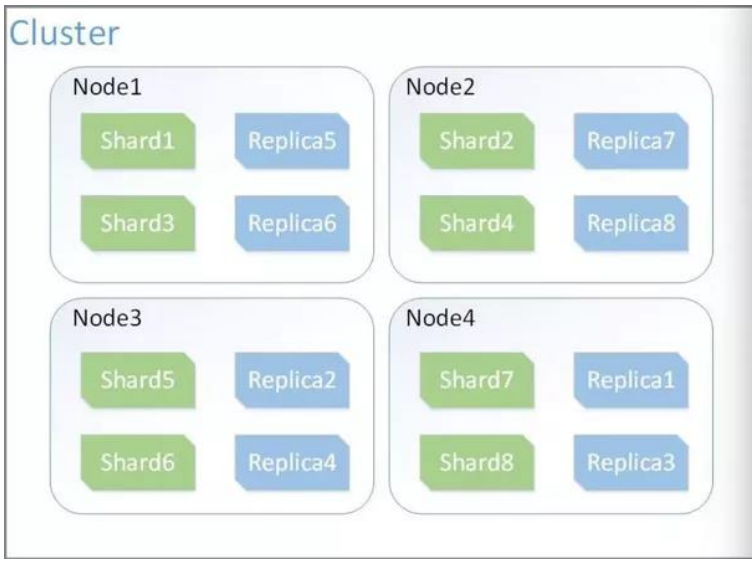
Elasticsearch expanderar på Lucene med att tillåta ett mera distribuerat system med multitenans (multitenancy). Detta hjälper systemet att skala sig för att klara av större arbetsbelastning som Big Data för med sig.

Distribuering av arbetet sker genom att dela upp indexen till fragment (shards). Fragment kan ha noll eller flera kopior (replicas) och är distribuerade över en eller flera noder. Dessa noder koordinerar sedan operationer mellan fragmenten. Detta system fungerar automatiskt. Ett kluster består således av en eller flera noder (se fig.2) (Elasticsearch BV 2018).

Elasticsearch har ett HTTP baserat webbgränssnitt och såsom Lucene, indexerar det helt-text data till dokument med textfält, i detta fall till JSON dokument.

Elasticsearch är baserad på öppen Java källkod och använder Apache licens.

Elasticsearch hade sin första utgivning 2010. Utvecklingen har varit hittills väldigt aktiv och versionen är vid 6.1 (December 2017).



Figur 2. Illustration av ett Elasticsearch kluster.

3.2 Logstash och Beats

Logstash är en programvara menad för att importera data, hantera och filtrera det, för att sedan skicka data vidare till andra ändpunkter, t.ex. Elasticsearch, Nagios eller enkelt till CSV-fil, eller t.o.m. bara text i konsolen.

Funktionerna i Logstash är uppdelad i tre funktioner:

1. Input: import av data.
2. Filter: hantering och filtrering av data.
3. Output: vart data ska skickas efter behandling.

Data som importeras kan i teorin vara vad som helst, men den är främst menad för data från loggfiler, metrik, webbapplikationer, molntjänster, etc. För närvarande finns det över 200 insticksprogram för inhämtning av data från olika källor. Denna information behandlas av Logstash och indexeras sedan i Elasticsearch (Elasticsearch BV 2018).

Denna behandling kan innebära t.ex. användning av GeoIP databaser för att spekulera geografiska koordinater på basis av IP adress och exkludering av känsliga data på basis av nyckelord. Logstash erbjuder flera färdiga filter och skapande av egna filter är också möjligt med hjälp av t.ex. Ruby kod (Elasticsearch BV 2018). Förutom att skicka behandlat data till Elasticsearch, är det också möjligt att skicka data till andra ändpunkter. Som exempel går det att skicka direkt till olika databaser, såsom MongoDB och InfluxDB. I detta arbete används Logstash endast för data som inte direkt kan importeras i Elasticsearch.

Beats är en nyare plattform skapad av Elasticsearch BV för skickande av loggfiler och metrik från tjänster. Plattformen är uppbyggd av så kallade Beats agenter (se tab.2), som installeras på källan (t.ex. server), därifrån man vill skicka data. Data från Beats agenter kan skickas till Logstash för behandling eller direkt till Elasticsearch för indexering. Beats agenterna är skapade för att vara lättviktiga i anseende till resursanvändning på datakällan. Detta åstadkoms delvis genom att agenterna endast minimalt behandlar data förrän det skickas. Plattformens största fördel är att skicka data direkt till Elasticsearch, därmed sparar man resurser eftersom data inte behöver skickas till och behandlas i Logstash och sedan indexeras i Elasticsearch (Elasticsearch BV 2018).

Tabell 2. Beats agenter

Filebeat	Skickar råa loggfiler.
Metricbeat	Samlar metrik från källan, såsom resursanvändning och servicestatistik.
Packetbeat	Skickar data och loggar gällande nätverksprotokoll. Tillåter monitorering av nätverk.
Winlogbeat	Skickar loggar från Windows operativsystem.

Auditbeat	Skickar loggar från Linux operativsystem som använder auditd (Linux Audit system).
Heartbeat	Enkel monitorering av tjänster (såsom webbsidor) för att kolla om de är online.

Utöver de officiella Beats agenterna i tabell 2, finns det också så kallade Community Beats, d.v.s. Beat agenter skapade av användare. Vid tidpunkten för skrivandet finns det 67 stycken av dessa. Som exempel finns det *wmibeat*, som kan skapa loggdata från Windows Media Instrumentation kommandon och Apachebeat, som specialiserar sig på att skicka loggdata från Apache webbserver.

3.3 Kibana

Kibana komponenten står för användargränssnittet och visualisering av data. Då man har en Elastic Stack instans installerad och i produktion kommer det främst att vara Kibana som används på daglig basis, via webbläsare. Kibana är i grunden en visualiserings plugin för Elasticsearch.
Kibana tillåter bl.a. skapandet av histogram, linjegrafer och pajdiagram på basis av data. Andra funktioner, såsom visualisering av data på geografisk basis är också möjligt.
Av dessa former av visualisering kan man skapa instrumentbräden, d.v.s. skräddarsydda vyer (se fig.3). Dessa vyer kan användas för att snabbt kunna se den relevanta informationen enligt användarens behov.

Kibana är skriven i JavaScript och publicerad med öppen källkod i form av Apache licens. Utvecklingen sker på Git och är väldigt aktiv, med versionsuppdateringar flera gånger i månaden. Januari 2018 är versionen vid 6.2.0.

Figur 3. Exempel på Kibana instrumentbräde

3.4 X-Pack

X-Pack är namnet på en serie programvaror som tillbringar utökad funktionalitet till Elastic Stack, som erbjuds av bolaget Elasticsearch BV. Det är inte ett obligatoriskt paket men det erbjuder flera uppskattade funktioner utöver Elastic Stacks inbyggda funktionalitet. X-Pack är utgiven som öppen källkod, dock till skillnad från de andra Elasticsearch produkterna krävs det avgiftsbelagd licens för att använda största delen av komponenterna i X-Pack (Elasticsearch BV 2018).

3.4.1 Security

Security paketet erbjuder funktionalitet för autentisering och auktorisering till Elastic Stack. Detta underlättar åtkomst till Elastic Stack och förbättrar säkerheten. Med hjälp av detta paket möjliggörs central autentisering via system såsom Microsoft Active Directory, LDAP (Lightweight Directory Access Protocol) och SAML (Security Assertion Markup

Language). Detta gör det möjligt för Elastic Stack användare att använda t.ex. sina Windows inloggningsuppgifter för att logga in i Kibana. Detta underlättar arbetet för administratören, som inte behöver skapa skilda användarkonton och kan tilldela rättigheter på basis av t.ex. Active Directory säkerhetsgrupper.
Utöver detta erbjuder också Security paketet SSL/TLS krypterad trafik mellan noder, HTTP och andra transport klienter. IP filtrering ingår också, som man kan använda för att begränsa inkommande och utgående trafik från klustret.
Till sist erbjuder Security paketet händelsegranskning, d.v.s. granskning av system- och användarhändelser. Detta gör det möjligt att t.ex. se när och var vem har tittat på data.

3.4.2 Alerting

Alerting paketet erbjuder funktionalitet för alarmering och notifikationer på basis av data i Elastic Stack. Man kan definiera olika trösklar för värden, där de registreras som alarm ifall de överskrids i Elastic Stack. Detta alarm kan sedan informeras via t.ex. e-post, Slack och PagerDuty. Det går också att koppla det till egen extern webbtjänst med webhooks.
Utöver alarmering via manuellt inställda trösklar så går det också att använda sig av maskininlärning för att ha Elastic Stack att själv känna igen ovanliga och stora förändringar i värden och sedan göra alarm av det. (Elasticsearch BV 2018)

3.4.3 Reporting

Med Reporting paketet kan man generera rapporter utifrån Kibana visualiseringar och instrumentbräden. Dessa rapporter har vyer optimerade för utskrift, är PDF formaterade och går att skräddarsy enligt egen smak t.ex. med logo. Rapporter går att generera vid behov, enligt tidtabell och på basis av dataförändringar i Elastic Stack.
Rapporter är främst menade för att kunna dela med utomstående, förmän, kunder, o.s.v. Det kan också vara av nytta för presentation av data där man inte har möjlighet att komma åt sin Elastic Stack på distans.
Utöver dessa användarvänliga rapporter går det också att skicka rapporter i CSV-form (comma-separated values), som är lättare att hantera av annan programvara (Elasticsearch BV 2018).

3.4.4 Graph

Graph modulen tillåter granskning och visualisering av kopplingar mellan data i Elasticsearch. Den möjliggör länkning av t.ex. personer, ställen, preferenser och produkter på basis av gemensamma data. Detta underlättar märkvärt utforskning av data från ett användarvänligt perspektiv.
Modulen specialiserar sig på att kategorisera data enligt nyckelord, t.ex. TCP/IP data som nätverksinformation, processoranvändning som resursförbrukning. Detta hjälper med att snabbt hitta relevant information vid sökningar. Exempel på scenarion där det kan vara till nytta av denna modul:

- Identifiera användare på basis av IP adress och kunna se resursförbrukning av dina tjänster.
- Hitta gemensam nämnare i diverse problem, såsom ovanlig resursförbrukning eller programfel.

Graph modulen har också ett eget API (applikationsprogrammeringsgränssnitt), som man kan använda för att hämta data till extern tjänst eller program. Detta API använder sig av Elasticsearch sökningssyntax (Elasticsearch BV 2018).

3.4.5 Machine Learning

Denna modul hjälper med att hitta avvikelser i data och därmed hjälper den administratören identifiera problem. Modulen fungerar genom att analysera inkommande data och bygga en modell av s.k. normal uppförande. Efter detta kan modulen identifiera avvikelser från detta uppförande och via t.ex. Alerting modulen skicka notis via e-post.
Exempel på användning:

- Identifiera drastisk minskning i resursförbrukning på en tjänst och därmed känna igen fel.
- Identifiera ökad trafikmängd till en tjänst som inte vanligtvis har det och därmed undersöka det.

Utöver detta erbjuder modulen också möjligheten att förutspå data. T.ex. om förbrukning av skivutrymme på en server ökar i en viss takt så kan det förutspås när skivan kommer att bli fylld av data (Elasticsearch BV 2018).

4 IMPLEMENTERING AV ELASTIC STACK

Planeringen av implementationen är viktig. Till detta hör val av plattform, hårdvaruplanering, storlek på kluster och indexbehandling. Elastic Stack är väldigt modulärt och lätt att utöka, därtill är få saker i konfigurationen permanenta och icke-modifierbara. Detta betyder att om man gjort fel i implementationen så är det relativt lätt att åtgärda. Det är dock rekommenderat att skaffa sig förståelse för systemet och först sedan implementera. Detta sparar tid och hjälper till att bygga upp ett effektivt system från första början (Gormley & Tong 2015 s. 631 ff).
Eftersom konfiguration av alla komponenter i Elastic Stack sker via filer, bestämdes det att använda sig av GIT för versionshantering av konfigurationerna.

4.1 Val av plattform

Då det gäller plattform så finns det flera val, eftersom både Elasticsearch och Logstash är programmerade i Java, som kan köras på alla plattformar som har stöd för det. Kibana är inte skriven i Java men finns kompilerat till Windows, MacOS och Linux. Utöver detta finns också alla Elastic Stack komponenter tillgängliga som officiella Docker paket (virtualiserad applikation). I detta fall valde vi med uppdragsgivaren att välja Linux Debian som serverplattform, främst på grund av dess stabilitet och minimala resursanvändning.

4.2 Hårdvaruplanering

Elasticsearch och Logstash är de mest krävande komponenterna i Elastic Stack. Båda komponenterna konsumerar mycket primärminne och prestandan riskerar minska ifall minnet inte räcker till på noden (Gormley & Tong 2015 s. 634). Elastic Stack använder också en del processorresurser, dock rekommenderas det mera kärnor istället för hög klockfrekvens. Till sist är det bra att ge noderna gott om skivutrymme och vara beredd att vid behov utöka det.

Det är svårt att uppskatta hur mycket varje nod kommer att behöva. Hur mycket skivutrymme som upptas beror på mängden data. Det påverkas på tre sätt, mängden data som matas in i Elasticsearch, hur länge man vill förvara indexen, och till sist hur många fragment man valt att använda för indexen. Det är bäst att vara beredd på att ge mera utrymme till noderna vid behov, eller ändra på sin strategi med indexen.
Som grund för noderna bestämde vi oss för följande med uppdragsgivaren (se tab.3).

Tabell 3. Grundspecifikationer för nod

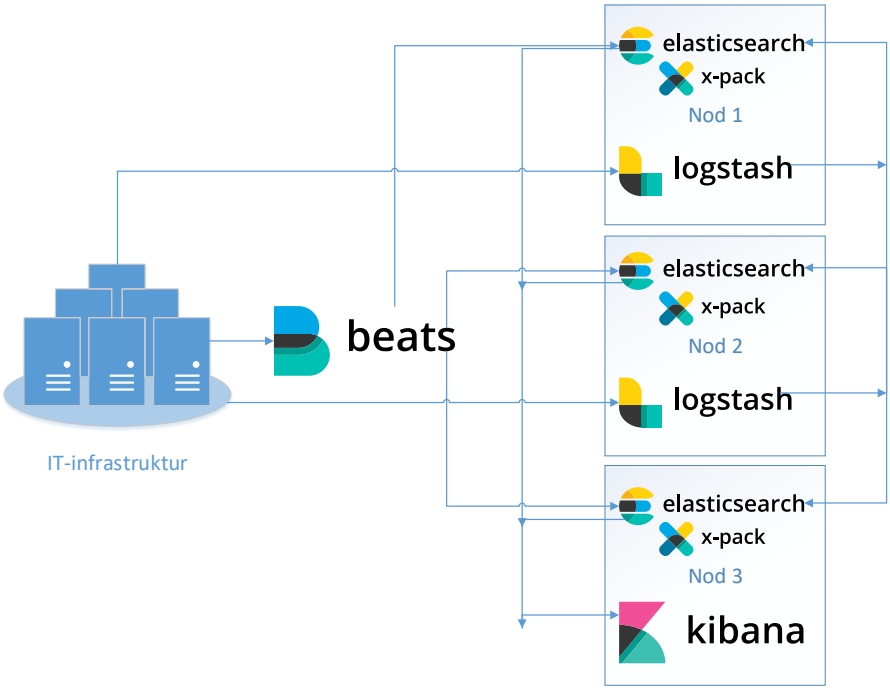
Processorkärnor	4 stycken
Centralminne	6192 megabyte
Skivstorlek	40 gigabyte systemskiva, 200 gigabyte dataskiva för Elasticsearch indexer

Vid behov kan dessa specifikationer ändras enkelt, eftersom noderna kör på virtuella maskiner.

4.3 Elastic Stack kluster

Ett Elastic Stack kluster behöver bestå av åtminstone en nod. För redundans och prestanda rekommenderas ändå fler noder. Om man har bara en nod så går man miste om data varje gång man behöver göra service åt noden. Två noder är inte heller att rekommendera. Detta beror på att man i ett Elasticsearch kluster behöver definiera minimimängd berättigade master noder som krävs för att forma ett kluster. Av dessa berättigade noder väljs en som master nod, som sedan bestämmer när nya index skapas och hur de fördelas. Om man med två berättigade master noder sätter denna inställning till ett, riskerar man en s.k. *split-brain* scenario, där båda noder anser sig själv vara master, som sedan leder till ett felfungerande kluster (Gormley & Tong 2015 s. 637). För denna orsak rekommenderas som minimi ett kluster på åtminstone tre noder.
Med uppdragsgivaren kom vi därmed överens om att grunda klustret med tre noder med de olika Elastic Stack rollerna uppdelade mellan dem. Tre noder kommer ha Elasticsearch installerat, två har Logstash och endast en nod har Kibana installerat (se fig.4).

Vad gäller sedan själva infrastrukturen som ska förse klustret med data, så är det primärt Beats som ska användas om det är möjligt, som klarar av att skicka data direkt till Elasticsearch. Endast vid behov kommer data från funktioner såsom fjärrsystemloggning och API förfrågningar skickas till Logstash för behandling, som sedan i tur skickar behandlade data till Elasticsearch klustret.



Figur 4. Illustration av Elastic stack kluster som implementeras

Vid behov kan klustret förstoras horisontellt genom att sätta upp fler noder. Alternativt kan man också förstora klustret vertikalt, d.v.s. öka prestandan på existerande noder genom att ge dem fler processorkärnor och/eller mera systemminne.

4.4 Installation av Elasticsearch, Logstash och Kibana

Skaparna av Elastic Stack har gjort det enkelt att installera komponenterna. De erbjuder färdiga binärpaket för MacOS, Windows och Linux. För Windows finns det MSI paket, och för Linux finns det .deb paket för Debian baserade distributioner och .rpm paket för Red Hat baserade. Utöver detta erbjuder de också officiella Docker avbildningar. För Linux distributioner är det dock att rekommendera pakethanterare såsom *apt* eller *yum*. För båda två finns det officiella datakataloger där man enkelt kan få den senaste versionen och i framtiden uppdatera installationen. (Elasticsearch BV 2018).

4.4.1 Installation av Elasticsearch

Eftersom Elasticsearch är skriven i Java så behöver man en Java virtuell maskin (JVM) för att kunna köra den. Enligt dokumentationen rekommenderas Java 8 Update 131 eller nyare versioner. Elasticsearch stöder både Oracle företagets och OpenJDKs JVM. OpenJDK baserar sig på öppen källkod. För denna orsak bestämde vi med uppdragsgivaren att välja det. Java installeras enkelt genom att först lägga till korrekt datakatalog och sedan installera det med apt. Efter detta så installerar man Elasticsearch på samma sätt, d.v.s. genom att lägga till datakatalogen och sedan installera Elasticsearch med apt. Efter installation så existerar det nu i systemet en service för Elasticsearch som kör under användarkontot *elasticsearch* som installeringen skapade.

4.4.2 Konfigurering av Elasticsearch

Konfigurering av Elasticsearch sker i huvudsak genom YAML filer (YAML Ain't Markup Language). Dessa existerar under katalogen */etc/elasticsearch*. Den huvudsakliga konfigurationsfilen är **elasticsearch.yml.** De viktigaste konfigurationerna som man ska ändra är illustrerade i tabell 4.

Tabell 4. Viktigaste konfigurationselementen i Elasticsearch

Konfigurationselement	Förklaring
cluster.name	Klusternamn. Behöver vara samma för alla noder som ska vara i samma kluster.
node.name	Nodens namn. Variabler kan användas, såsom ${HOSTNAME}.
path.data	Där indexen sparas. Bör finnas tillräcklig med lagringsutrymme.
network.host	IP adress som Elasticsearch binder sig till. 0.0.0.0 kan användas för alla nodens IP adresser.
discovery.zen.ping.unicast.hosts	IP adress eller DNS namn för alla noder i klustret.
discovery.zen.minimum_master_nodes	Minimum mängd valbara master noder som krävs för att klustret ska funktionera.

Utöver elasticsearch.yml konfigurationsfil så finns det också filen **jvm.options** där man kan konfigurera inställningar för JVM, som Elasticsearch kör på. Överlag rekommenderas det inte att ändra på standardinställningarna, förutom en. Denna inställning är JVM *heap size*. Denna ställer in hur mycket minne man tillåter Elasticsearch att ta nytta av. Standardinställningen är konservativt inställd till 1 GB. Rekommendationen är att ställa in denna till hälften av systemminnet som noden har tillgång till (Gormley & Tong 2015 s. 641 f.). I detta fall har noderna tillgång till 6 GB minne, vilket enligt rekommendationen betyder att heap size borde ställas till 3GB. Detta görs genom att ställa in alternativen **–Xms** (minimum) och **–Xmx** (maximum) till –Xms3g och –Xmx3g.

4.4.3 Installation av Logstash

Installationen av Logstash sker på samma sätt såsom Elasticsearch. Logstash är skriven i Java och behöver därmed också en JVM installerad. Logstash finns på samma datakataloger som Elasticsearch. Om man därmed har installerat Elasticsearch från tidigare så kan man enkelt installera Logstash med *apt*.

4.4.4 Konfigurering av Logstash

Logstash konfiguration sker via modifiering av filer under */etc/logstash*. Filen **logstash.yml** är den huvudsakliga. Där konfigureras egenskaper såsom var loggfiler tillfälligt sparas, hur många trådar ska användas för varje data-pipeline. I allmänhet behöver man inte modifiera dessa inställningar. Den viktigaste konfigurationen sker för varje pipeline där man tar in data till Logstash. Sedan Logstash 6.0 finns det stöd för multipla pipeliner. Före det var man tvungen att använda antingen en konfugrationsfil eller köra flera instanser av Logstash (João Duarte 2017). I filen **pipelines.yml** definierar man ID och stället för konfigurationsfilen för varje pipeline (se fig.5).

```
- pipeline.id: apache
    path.config: "/etc/logstash/conf.d/apache.conf"

- pipeline.id: elastiflow
    path.config: "/etc/logstash/elastiflow/conf.d/*.conf"
```

Figur 5. Exempel på Logstash configuration

Konfigurationen av dessa moduler behöver alltså göras för inmatning av loggfiler. Detta behöver göras eftersom loggfiler ser olika ut beroende på källan (Elasticsearch BV 2018).

4.4.5 Exempelkonfiguration av Logstash modul

Konfiguration av en pipeline går ut på tre delar: Inmatning, filtrering och utmatning. I input delen definierar man protokoll (TCP eller UDP), portar och typ av data. Om man kör flera pipeliner ska man se till att porten som man tar in data på är unik (se fig.6).

```
input {
  tcp {
    port => 5000
    type => syslog
  }
  udp {
    port => 5000
    type => syslog
  }
}
```

Figur 6. Exempel på Logstash inmatningskonfiguration

Filtrering sker genom att använda olika instickspogram, av vilka **grok** är den mest mångsidiga. Den tillåter användning av reguljära uttryck för att parsa data, lägga det till fält enligt val och ytterligare modifiera det om man så vill (se fig.7).

```
input {
  file {
    path => "/var/log/http.log"
  }
}
filter {
  grok {
    match => { "message" => "%{IP:client} %{WORD:method}
    %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}" }
  }
}
```

Figur 7. Exempel på Logstash grok filter

I utmatningsdelen definierar man vart data ska skickas efter behandling i Logstash. I detta fall vill vi skicka det till Elasticsearch. Det viktigaste att definiera här är till vilka noder

man vill skicka data (med lastbalansering) och hur indexet definieras. I detta fall definierar vi alla noder vi har i klustret och ger eget namn till indexen med datum. Vid utveckling och testning med Logstash kan det vara av nytta dock att använda *stdout* utmatning, d.v.s utmatning i terminalen. På detta vis kan man stegvis utveckla sin pipeline för att få data att se ut som man vill (se fig. 8).

```
output {
    elasticsearch {
    hosts => ["elk-node-1:9200", elk-node-2:9200, elk-node-3:9200]
    index => "logstash-apache-%{+YYYY.MM.dd}"
    }
}
```

Figur 8. Exempel på Logstash utmatning till Elasticsearch

4.4.6 Installation av Kibana

Installation av Kibana går till på samma sätt de andra produkterna i Elastic Stack, dock kräver Kibana inte en JVM för att fungera. Installation på Linux sker enklast genom att sätta till Elasticsearchs officiella datakatalog och sedan installera Kibana med *apt*.

4.4.7 Konfigurering av Kibana

Grundkonfigurationen av Kibana görs i **kibana.yml**. I denna fil är det viktigt att konfigurera egenskaperna **server.host** och **elasticsearch.url**. Server.host bestämmer på vilken IP adress Kibana kör. Man kan ställa denna in på 0.0.0.0, då lyssnar Kibana på inkommande trafik på alla dess IP adresser, dock har en server oftast endast en IP adress. Elasticsearch.url bestämmer till vilken Elasticsearch nod Kibana skall skicka Elasticsearch anrop. Denna inställning bör sättas till en nod med koordinations kapabilitet som kan genom lastbalansering fördela Elasticsearch anrop mellan alla noder med data. Alla master valbara noder har denna förmåga, dock rekommenderar Elasticsearch BV att installera en separat koordinerande nod ifall man har ett stort kluster eller om man märker att sökningar i Elasticsearch tar långa tider.

4.5 Installation av Beats agenter i infrastrukturen

De olika Beats agenterna som finns tillgängliga behöver installeras på varje server som man vill bevaka via Elastic Stack. Beats finns tillgängligt för Windows, Linux och MacOS. Beroende på tjänster och vad man vill bevaka, behöver man välja ut vilka Beats agenter man vill använda och med vilken konfiguration. T.ex. på en Linux server som kör Apache kan det vara ändamålsenligt att installera Metricbeat för att bevaka systemresurser med Metricbeats system-modul, medan Apache-modulen kan användas för att noggrannare kontrollera Apache metrik. Utöver detta kan man också använda sig av Filebeat för att skicka loggfiler från servern antingen till Logstash för ytterligare modifiering/berikning av data eller direkt till Elasticsearch (Elasticsearch BV 2018).
Installation av Beats på Windows sker genom att ladda ner paketet från Elasticsearchs webbsidor och sedan köra ett inkluderat Powershell skript. Detta installerar en service på systemet, som man kan ställa in på automatisk start av datorn.
På Linux kan man installera Beats genom att använda pakethanterare. Datakatalogen som bör användas är samma som för de andra produkterna i Elastic Stack.

4.5.1 Installation och konfiguration av Metricbeat

Metricbeat konfigureras i filen **metricbeat.yml**, där man kan ställa in optioner såsom mängden fragment i Elasticsearch indexet, inladdning av standard översikter i Kibana och vart metriken skall skickas. Sedan kan man konfigurera de olika modulerna man vill använda i *modules.d* katalogen. Beroende på modulen så finns det diverse optioner som kan konfigureras, t.ex. i system-modulen kan man bestämma vilka resursmetriker ska skickas, t.ex. processor- och minnesanvädning. I alla moduler kan man definiera hur ofta metrik skall samlas och skickas. Ett exempel illustreras i figur 9.

```
module: system
period: 10s
metricsets:
  - cpu
  - memory
  - network
  - process
  - process_summary
processes: ['.*']
process.include_top_n:
  by_cpu: 5      # include top 5 processes by CPU
  by_memory: 5   # include top 5 processes by memory

module: system
period: 1m
metricsets:
  - filesystem
  - fsstat
processors:
- drop_event.when.regexp:
    system.filesystem.mount_point: '^/(sys|cgroup|proc|dev|etc|host|lib)($|/)'

module: system
period: 15m
metricsets:
  - uptime
```

Figur 9. Exempel på Metricbeat modulens konfiguration

För att aktivera en modul så skall man se till att *.disabled* delen i modulens filnamn är borttaget. Efter detta kan man starta Metricbeat tjänsten.

4.5.2 Installation och konfiguration av Filebeat

Filebeat konfigureras generellt på samma sätt som Metricbeat, d.v.s i **filebeat.yml** och sedan vid behov i de skilda modulernas konfigurationsfiler. I huvudsak rekommenderas det att använda de färdiga modulerna p.g.a att de är färdigt optimerade för att forma data lämpligt för att direkt kunna skickas in i Elasticsearch, istället för till Logstash för behandling. T.ex. för systemloggar kan man använda systemmodulen. För loggfiler som inte har någon lämplig modul så behöver man ställa in en skild konfiguration i filebeat.yml. Konfigurationen delas in i olika s.k. prospektorer där man åtminstone behöver definiera typ av fil (t.ex. logg) och ställe var filerna finns.

En begränsning med Filebeat är att det inte går att definiera multipla utmatningar av data. Man är tvungen att välja en, t.ex. Logstash eller Elasticsearch. Beroende på serven så

finns det situationer där det skulle vara ändamålsenligt att dela upp utmatningen, t.ex. skicka systemloggar direkt till Elasticsearch, medan man skickar loggfiler för en annan applikation till Logstash för behandling. Det enda sättet att få denna konfiguration är att installera två instanser av Filebeat på servern (Elasticsearch BV 2018). Ett exempel på modulkonfiguration kan ses i figur 10.

```
module: mysql
# Error logs
error:
  enabled: true

  # Set custom paths for the log files. If left empty,
  # Filebeat will choose the paths depending on your OS.
  #var.paths:

# Slow logs
slowlog:
  enabled: true

  # Set custom paths for the log files. If left empty,
  # Filebeat will choose the paths depending on your OS.
  #var.paths:
```

Figur 10. Exempel på konfiguration av Filebeat MySQL modul

4.5.3 Installation och konfigurering av Winlogbeat

Winlogbeat är endast avsedd för Windows operativsystem och fungerar genom att läsa och skicka händelser vidare från Windows loggbok till valbar utgång (t.ex. Logstash eller Elasticsearch). Installationen sker på samma sätt som Metricbeat och Filebeat, d.v.s. genom att ladda ner paketet från Elasticsearchs webbsidor och sedan köra det inkluderade Powershell skriptet för att installera programmet som en tjänst. Konfiguration av Winlogbeat sker i filen **winlogbeat.yml**. I filen kan man konfigurera vilka Windows loggar skall skickas och vart de skall skickas. T.ex. kan man välja att skicka endast Applikationsloggar om man så vill. Effektiv filtrering av händelserna är också möjlig. Man kan bl.a. minska mängden exporterade fält och välja att lämna bort alla händelser av typen information, men inte av typerna varning eller fel. Man kan också göra avancerad filtrering genom reguljära uttryck i de olika fälten i händelserna. Detta är att rekommendera för att slippa mängden redundant data i Elasticsearch.). Ett exempel på modulkonfiguration av Winlogbeat kan ses i figur 11.

```
winlogbeat.event_logs:
  - name: Application
    ignore_older: 72h
  - name: Security
  - name: System

setup.template.settings:
  index.number_of_shards: 3

processors:
- drop_event:
    when.or:
      - equals.log_name: 'Information'

output.elasticsearch:
  hosts: ["elk-node-1", "elk-node-2", "elk-node-3"]
```

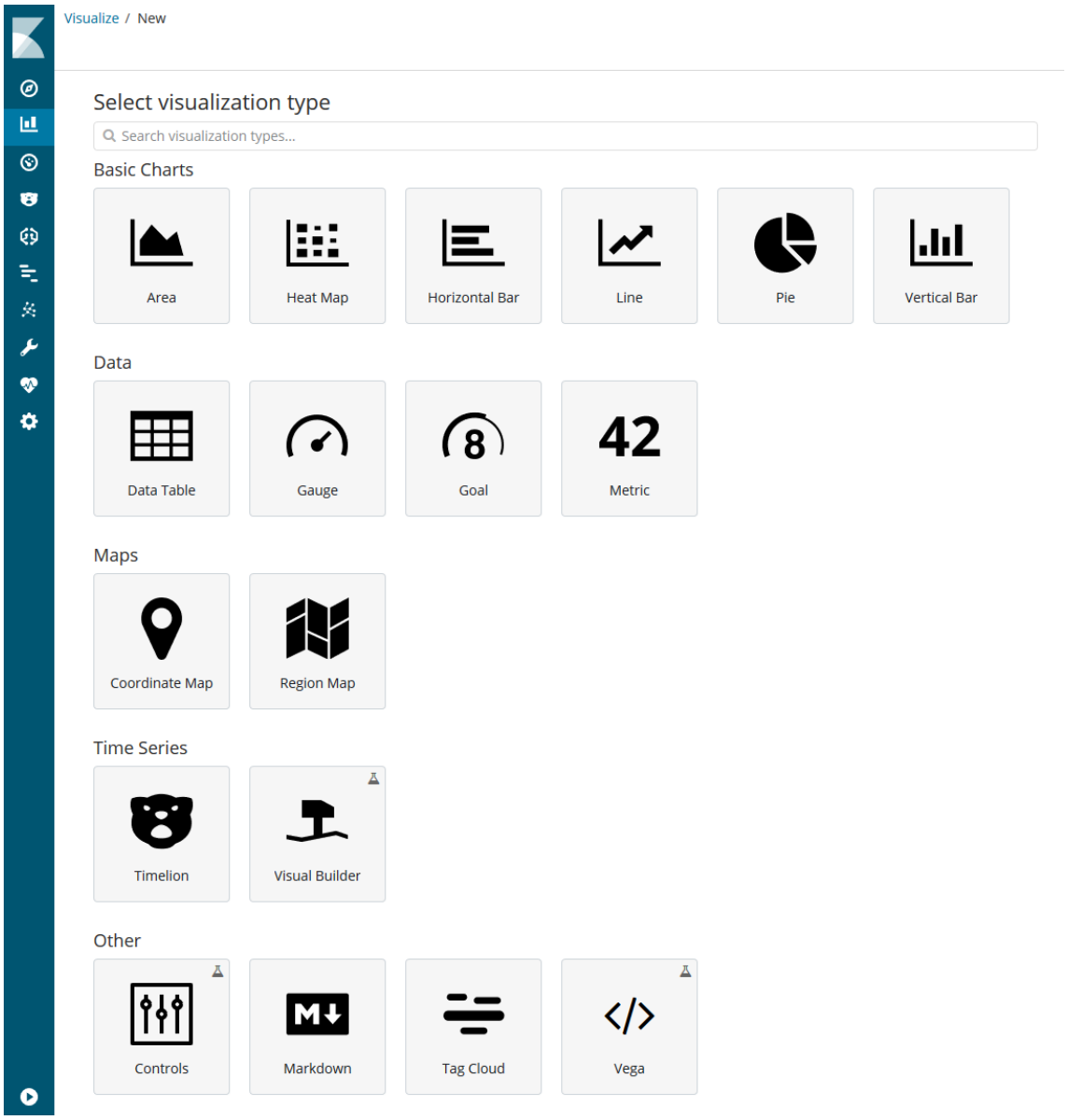
Figur 11. Exempel på Winlogbeat konfiguration

4.6 Visualiseringar i Kibana

Kibana stöder flera olika visualiseringar som man kan sedan kombinera till diverse s.k. instrumentbrädor. Man kan snabbt och enkelt följa med data som man valt att se och borra in sig djupare i data vid behov. Elastic stack har också flera färdiga mallar som man kan använda som instrumentbrädor.

4.6.1 Skapande av visualisering

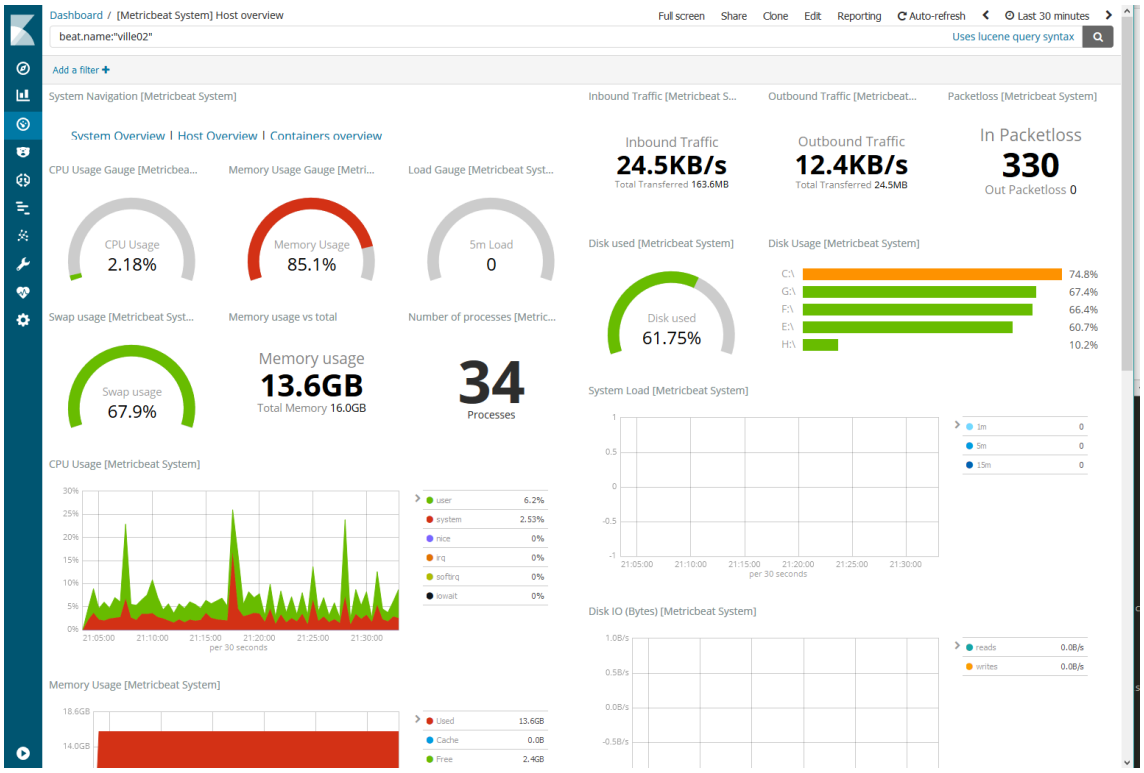
Skapande av egen visualisering börjar genom att man går in i *Visualize* delen av Kibana. Sedan skall man välja typen visualisering. Det finns många att välja mellan, t.ex. histogram, linjegrafer och pajdiagram. Utöver detta finns det också mera avancerade visualiseringar, såsom *Vega* grafer, där man kan använda sig av **D3.js** visualiseringsbiblioteket för att bygga fullständigt skräddarsydda visualiseringar (Elasticsearch BV 2018).
Då man valt typen av visualisering och index, behöver man välja form av *aggregation*. Detta kan vara t.ex. medeltal, minimum, maximum eller summa. Detta uträknas på basis av vilken tidsperiod man valt i Kibana. Sedan väljer man fält som man vill räkna ut detta på. Till sist kan man gruppera resultaten via gemensamma data, t.ex. servernamn (se fig.12).



Figur 12. Former av visualiseringar som kan skapas i Kibana

4.6.2 Skapande av instrumentbräde

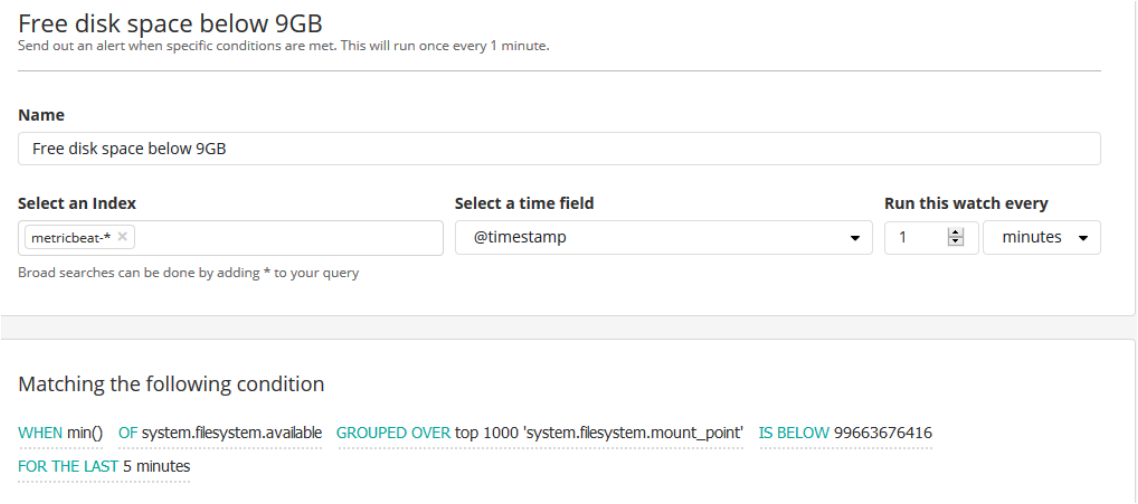
Ett instrumentbräde består helt enkelt av olika visualiseringar (se fig.13). Man importerar visualiseringar till instrumentbrädet, där man sedan kan modifiera utseendet på visualiseringen. Det går t.ex. att ändra storlek på visualiseringen och färgerna på elementen. Det är viktigt att komma ihåg att data som representeras är alltid ihopkopplat med tidsperioden som är vald i Kibana.



Figur 13. Exempel på Kibana instrumentbräde

4.7 Alarmering

Alarmering sker enklast genom att använda sig av Elasticsearchs officiella Watcher tilllägg, som är del av X-Pack (se fig.14). För att installera Watcher behöver man installera X-Pack paketet. Detta sker genom att köra kommandot:
/usr/share/elasticsearch/bin/elasticsearch-plugin install x-pack.
Efter detta hittar man Watcher modulen i Kibana under Management->Elasticsearch->Watcher. Principen med Watcher module är att om det går att söka, så går det att göra alarmering om det. Watcher går alltså ut på att på att definiera Elasticsearch sökningar och om det blir en träff bestämmer man en handling. För tillfället går det att definiera tre olika handlingar i Watchers grafiska gränssnitt; logg, e-post och meddelande i Slack chatsystemet.



Figur 14. Skapande av alarm i Kibana

För mera avancerade funktioner såsom sökningar med aggregationer, jämförelse av data, handlingar med ytterligare stöd för bl.a. webhooks, hipchat och pagerduty, kan man skapa konfigurationer i JSON format (se fig.15). Efter att man skapat en konfiguration för en Watcher behöver man ta den i bruk via ett HTTP PUT kommando.

```
PUT _xpack/watcher/watch/log_error_watch
{
  "trigger" : { "schedule" : { "interval" : "10s" }},
  "input" : {
    "search" : {
      "request" : {
        "indices" : [ "logs" ],
        "body" : {
          "query" : {
            "match" : { "message": "error" }
          }
        }
      }
    }
  },
  "condition" : {
    "compare" : { "ctx.payload.hits.total" : { "gt" : 0 }}
  },
  "actions" : {
    "log_error" : {
      "logging" : {
        "text" : "Found {{ctx.payload.hits.total}} errors in the logs"
      }
    }
  }
}
```

Figur 15. Konfigurat av alarm i JSON format

4.8 Underhåll av Elasticsearch indexer med Curator

Med tiden kan Elasticsearch indexer bli väldigt stor, beroende på hur ofta och hur många ändpunkter man har som skickar data. På fyra dagar, med 35 index och lite över 5 miljoner dokument ihopsamlades 3GB data på varje nod. P.g.a. att vi definierat tre fragment för varje index finns det också tre fragmentkopior. Detta betyder att fragmenten tar upp dubbelt sin skivstorlek (Gormley & Tong 2015 s. 28).

Curator är en programvara menad för underhållning av Elasticsearch index. Med den kan man bl.a. slå ihop index (forcemerge), minska mängden fragment (shrink), radera index (delete) och även skapa nya index på basis av ålder, mängd dokument eller index storlek (rollover) (Elasticsearch BV 2018).

Syn-Hershko (2016) rekommenderar en konfiguration för hantering av index med följande process:

1. Minska mängden fragment till ett fragment för varje index.
2. För kompatibla index, slå ihop dem till en.
3. Efter en viss tid, ta bort indexen.

Detta kan man konfigurera att köra t.ex. veckovis, d.v.s. alla föregående veckans index slås ihop till ett.

Ett alternativ till detta är att använda sig av rollover strategin och samtidigt förenkla index principen. Enligt standardinställning skapas det nya index varje dag via Logstash och Beats. Istället för detta kunde man skriva till ett och samma index och vid behov göra en s.k. rollover med Curator på basis av bestämda omständigheter, såsom indexålder, dokumentmängd eller storlek. Denna strategi rekommenderas av Mildenstein (2018), underhållaren av Curator. Om man väljer denna strategi finns det skäl att ha namnformen *indexnamn-000001* på indexen, då inkrementerar Curator detta och gör en rollover till det nya indexet *indexnamn-000002*. För att förenkla konfiguration så kan man skapa en alias till indexen, t.ex. *indexnamn-000001* kunde ha alias *indexnamn*.

Ett exempel på Curator konfiguration där rollover tekniken används ses i figur 16.

```
  action: rollover
  description: >-
  Rollover index if larger than 5GB to a new index incremented by one.
options:
  name: metricbeat
  conditions:
    max_size: 5gb
  extra_settings:
    index.number_of_shards: 1
    index.number_of_replicas: 1
  timeout_override:
  continue_if_exception: False
  disable_action: False
```

Figur 16. Exempel på rollover Curator konfiguration

Utöver denna rollover strategi kan man också radera index t.ex. på basis av ålder på index eller storlek på index. Att göra det på basis av storlek förenklar administration, eftersom det då inte finns fara för att skivutrymmet skulle ta slut på klustret. Detta kan åstadkommas enkelt med Curators *delete_indices* funktion i samband med filtertyperna *pattern* och *space* (Mildenstein 2018). Mönsterfiltret baserar sig på indexets namn. Med *prefix* typen av mönster och värdet *metricbeat-* väljs endast alla metricbeat index. Storleksfiltret räknar storleken på indexen definierade med mönsterfiltret. Filtret adderar ihop index en och en på basis av ålder. Då den definierade tröskeln överskrids, väljs de föregående indexen för radering. Ett exempel på denna konfiguration ses i figur 17.

```
action: delete_indices
description: >-
  Delete indices when their cumulative size is larger than 50GB.
options:
  ignore_empty_list: True
- filtertype: pattern
  kind: prefix
  value: metricbeat-
- filtertype: space
  disk_space: 50
  use_age: true
  source: creation_date
```

Figur 17. Exempel på radering av index baserad på storlek

4.8.1 Installation, körning och schemaläggning av Curator

Curator installeras genom att lägga till Curators officiella datakatalog i operativsystemet. Sedan installerar man Curator med kommandot *apt install elasticsearch-curator*. För redundans rekommenderas det att installera Curator på varje master valbar nod. I konfigurationsfilen definierar man sedan Curator att köra endast på lokala noden, och *endast* om noden är master, detta görs genom att ställa in inställningen *master_only* till *True*. På detta sätt undviker man möjliga konflikter då Curator körs, samtidigt som man behåller redundans.

Körningen av Curator görs genom att starta programmet med parametrar. En modell för detta är följande, *curator [--config CONFIG.YML] ACTION_FILE.YML*. Själva schemaläggningen av körningen av Curator kan man hantera med operativsystemets inbyggda schemaläggare. I Linux operativsystem kan man sköta schemaläggningen med det inbyggda programmet Cron. Detta program går ut på att man i en fil, *Crontab*, definierar scheman och vad som ska köras. Syntaxen i Crontab är relativt enkel, eftersom schemat kan definieras på minut, timme och datum (se fig.18). Utöver detta finns det också förbestämda definitioner, såsom *@hourly, @daily, @weekly*, o.s.v (Pantz 2007).

```
# Minute   Hour   Day of Month       Month          Day of Week       Command
# (0-59)  (0-23)     (1-31)    (1-12 or Jan-Dec)  (0-6 or Sun-Sat)
    0        2          12              *                *           /usr/bin/find
```

Figur 18. Exempel på Crontab

Eftersom data matas in i Elasticsearch jämt, finns det det skäl att köra Curator ofta. I detta fall bestämdes det att köra Curator en gång i timmen. Konfigurationen i Crontab ser man i figur 19.

```
@hourly curator --config /etc/curator/config.yml /etc/curator/action_file.yml
```

Figur 19. Curator Crontab konfiguration

5 UTVÄRDERING OCH SAMMANFATTNING

Sökning av loggfiler på ett centraliserat ställe har många fördelar. När fel uppstår, går det snabbt att gå in på Kibana och göra en sökning på problemstället. T.ex. då en webbtjänst

gav HTTP felmeddelande 503 går det snabbt i Kibana att göra en sökning på det. I felloggarna kan man sedan identifiera orsaken till felet. Problemet kan sedan lösas med t.ex. omstart av webbtjänsten. För snabbare respons i framtiden kan man installera en alarmering nästa gång detta händer.
Belastning i nätverket går också enkelt att se i visualiseringar i Kibana och topplistan på förbrukare av trafiken.

5.1 Feedback av IT-teamet

Feedback av IT-teamet var positivt (Wikholm 2018). Det har länge varit behov för ett centraliserat system för logghantering och monitorering. Ett motsvarande kommersiellt liknande system med namnet *Splunk* hade utvärderats tidigare men till sist bedömdes det att vara för dyr lösning. Användargränssnittet bedömdes vara enkel att använda, instrumentbrädor gick snabbt att hitta och sökningar var enkla att göra. Användningen av Kibana dokumenterades också i uppdragsgivarens Atlassian Confluence Wiki system.
Existensen av ett logghantering- och monitoreringssystem upplevdes också att ge en bättre känsla av överblick på infrastrukturen. System såsom avbrottsfri kraftförsörjning var tidigare svåra att felsöka och kontrollera loggar på. Med det nya systemet kan man vid en sökning snabbt få överblick av loggarna.

5.2 Systemet i framtiden

Uppsättningen av Elasticsearch klustret kräver en hel del forskning för korrekt uppsättning. Konfiguration via filer är mångsidigt, dock blir det mycket jobb att hantera ju fler noder och klienter det finns. I detta avseende är det att rekommendera ett centraliserat konfigurationshanteringssystem, såsom *Ansible* för att enkelt kunna uppdatera konfigurationer. Vid tidpunkten för skrivande av detta arbete fanns det inget sådant system, dock kom man överens över att sätta upp ett sådant system i framtiden.
Utöver detta finns det mycket man kan ännu utveckla på. Vid skrivandet av detta arbete var endast en bråkdel av infrastrukturen kopplat till Elastic Stack. I framtiden ska fler nuvarande system konfigureras för att skicka sina loggar till Elastic Stack, likaså för nya tjänster som installeras i infrastrukturen.

Rapportering är också ett önskemål som kom fram. Elastic stack erbjuder rapportering via en modul i *X-Pack*. Detta utvärderades inte i arbetet, dock är det planerat att det i framtiden skall tas i bruk.
Machine Learning komponenten i X-Pack är också ett verktyg, som planeras att evalueras i framtiden. På papper ser den ut att erbjuda mångsidiga funktioner i samband med felsökning. Att snabbt hitta korrelationer och orsaker till problemfall är en värdefull funktion. Detta hjälper styra systemadministrationsarbetet till den proaktiva strategin, genom att istället för att lösa problem, försöker man lösa orsaken till problemen.

5.2.1 Sammanfattning

Elastic Stack visade sig att till stor del uppfylla uppdragsgivarens mål. Sökning av loggar på ett centraliserat ställe klassades som en ovärderlig resurs. Sökningarna i Elasticsearch upplevdes snabba och resultaten enkla att tolka. Instrumentbräden och visualiseringar i Kibana gav en bra överblick över de diverse system i infrastrukturen.
Den mest utmanande delen i implementationen upplevdes bestå av konfigurationen av Logstash. För att effektivt kunna behandla inkommande data behöver man behärska reguljära uttryck med instickspprogrammet *grok* (Ramati 2016), vilket kräver att man lär sig en ny form av syntax.
En av de största fördelarna med Elastic Stack är dess grund på öppen källkod. Utvecklingen sker snabbt och öppet. Vid problemfall upplevdes det enkelt att på diskussionsforum fråga om saken. Vid misstanke om fel i systemet kan man också skapa en *issue* på Elasticsearch GITHub sidor, där man överlag också kan följa med utvecklingen. Vid skrivandet av detta annonserade Elasticsearch BV öppnandet av källkoden till de stängda X-Pack komponenterna. Företaget ser sin framtid i öppen källkod (Banon 2018).
Sammanfattningsvis visade sig Elastic Stack vara en effektiv lösning till att hjälpa arbetsgivaren uppnå ett mera proaktivt sätt att arbeta med systemadministration. Detta hjälper IT-teamet att bättre kunna fokusera på utvecklingsarbete, dokumentation och upprätthållning istället för att reaktivt lösa problem.

KÄLLOR

Banon, Shay. 2018, *Doubling Down on Open*. Hämtad 29.4.2018 från https://www.elastic.co/blog/doubling-down-on-open

Carpenter, Jenny. 2014, *The 5 biggest challenges for today's IT admins.* Hämtad 13.02.2018 från https://www.solarwindsmsp.com/blog/5-biggest-challenges-for-it-admins

Cordray, Robert. 2015, *Why log management is absolutely critical for IT security*. Hämtad 14.2.2018 från https://www.itworldcanada.com/blog/why-log-management-is-absolutely-critical-for-it-security/377711

Dragland, Åse. 2013, *Big Data, for better or worse: 90% of world's data generated over last two years.* Hämtad 13.2.2018 från https://www.sciencedaily.com/releases/2013/05/130522085217.htm

Elasticsearch BV. 2018, *Beats Product Page.* Hämtad 13.2.2018 från https://www.elastic.co/products/beats

Elasticsearch BV. 2018, *Elasticsearch Product Page.* Hämtad 13.2.2018 från https://www.elastic.co/products/elasticsearch

Elasticsearch BV. 2018, *Elasticsearch Reference [6.2]*. Hämtad 13.2.2018 från https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started.html

Elasticsearch BV. 2018, *Kibana Product Page.* Hämtad 13.2.2018 från https://www.elastic.co/products/kibana

Elasticsearch BV. 2018, *Logstash Product Page.* Hämtad 13.2.2018 från https://www.elastic.co/products/logstash

Elasticsearch BV. 2018, *The Open Source Elastic Stack.* Hämtad 14.2.2018 från https://www.elastic.co/products

Elasticsearch BV. 2018, *X-Pack.* Hämtad 13.2.2018 från https://www.elastic.co/products/x-pack

Gormley, Clinton; Tong Zachary. 2015, *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine,* O'Reilly Media

João Duarte. 2017, *Introducing Multiple Pipelines in Logstash*. Hämtad 21.4.2018 från https://www.elastic.co/blog/logstash-multiple-pipelines

Jonathan W. Palmer. 2002, Web site usability, design, and performance metrics, *Information Systems Research Volume 13, Issue 2,* University of Maryland

Lindström, Karin. 2018, *Här är it-chefens största utmaningar under 2018.* Hämtad från https://cio.idg.se/2.1782/1.695923/cio-utmaningar

McCandless, Michael; Hatcher, Erik; Gospodnetić, Otis. 2010, *Lucene in Action, Second Edition. Manning, 457 s.*

MG Siegler. 2010, *Twitter Quietly Launched A New Search Backend Weeks Ago*. Hämtad 14.2.2018 från https://techcrunch.com/2010/10/06/new-twitter-search/

Mildenstein, Aaron. 2018, *Curator rollover with max_size.* Hämtad 29.4.2018 från https://discuss.elastic.co/t/curator-rollover-with-max-size/129966/5

Mildenstein, Aaron. 2018, *How can I name Curator rollover new Index like date prefix-YYYY.MM.DD-1*?. Hämtad 20.4.2018 från https://discuss.elastic.co/t/how-can-i-name-curator-rollover-new-index-like-date-prefix-yyyy-mm-dd-1/128542/3

More, Josh. 2016, *Breaking into Information Security: Crafting a Custom Career Path to Get the Job You Really Want,* Syngress Publishing

Pantz. 2007, *Cron and Crontab usage and examples.* Hämtad 29.4.2018 från https://www.pantz.org/software/cron/croninfo.html

Ramati, Ran. 2016, *A Beginner's Guide to Logstash Grok*. Hämtad 29.4.2018 från https://logz.io/blog/logstash-grok/

Syn-Hershko, Itamar. 2016, *All you need to know about Elasticsearch 5.0: Index management.* Hämtad 20.4.2018 från https://code972.com/blog/2016/11/102-all-you-need-to-know-about-elasticsearch-5-0-index-management

Thomas A. Limoncelli. 2017, *The Practice of System and Network Administration: Volume 1: DevOps and other Best Practices for Enterprise IT 3rd Edition.*, Addison-Wesley Professional

Wikholm, Mats. 2018, *Feedback gällande Elastic stack [muntl.]*, 26.4.2018

BILAGOR

Java installation.

```
# Add repo.
echo -e "\n# jessie backports\ndeb http://http.debian.net/debian jessie-backports main" >> /etc/apt/sources.list

# Install Java
apt update && apt install -t jessie-backports openjdk-8-jre-headless
```

Elasticsearch installation

```
# Add keys to apt
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key add -

# Install apt https support
apt install apt-transport-https

# Add Elastic repo
echo "deb https://artifacts.elastic.co/packages/6.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-6.x.list

# Install Elasticsearch
apt update && apt install elasticsearch

# Make sure elasticsearch has rights to data folder defined in config
chgrp -R elasticsearch /mnt/data/elasticsearch
chmod -R g+rwx /mnt/data/elasticsearch

# Set Elasticsearch service to start automatically at boot
update-rc.d elasticsearch defaults 95 10
```

Logstash installation

```
## Install Logstash

# Install
apt update && apt install logstash

# Set Logstash service to start automatically at boot
update-rc.d logstash defaults 95 10
```

```
# Install/update some codecs
./usr/share/logstash/bin/logstash-plugin install logstash-codec-sflow
./usr/share/logstash/bin/logstash-plugin update logstash-codec-netflow
```

Kibana installation

```
## Install Kibana

# Install
apt update && apt install kibana

# Set Kibana service to start automatically at boot
update-rc.d kibana defaults 95 10

# Set iptables to forward port 80 to 5601, save to file and set it inter-
face config
iptables -A INPUT -i eth0 -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --dport 5601 -j ACCEPT
iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 80 -j REDIRECT --to-
port 5601
iptables -t nat -I OUTPUT -p tcp -o lo --dport 80 -j REDIRECT --to-ports
5601
iptables-save > /etc/iptables.rules

echo -e "\t\tpre-up iptables-restore < /etc/iptables.rules" >> /etc/net-
work/interface
```

X-pack installation

```
## X-Pack

# Elasticsearch
/usr/share/elasticsearch/bin/elasticsearch-plugin install x-pack

# Enable/disable modify at /etc/elasticsearch/elasticsearch.yml

# Kibana
/usr/share/kibana/bin/kibana-plugin install x-pack

# Logstash
/usr/share/logstash/bin/logstash-plugin install x-pack
```

Curator installation

```
# Install key
wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | sudo apt-
key add -

# Add repo
echo "deb [arch=amd64] https://packages.elastic.co/curator/5/debian sta-
ble main" | sudo tee -a /etc/apt/sources.list.d/curator-5.x.list

# Install Curator
sudo apt-get update && sudo apt-get install elasticsearch-curator
```

Curator actionfile

```
actions:
  1:
    action: rollover
    description: >-
      Rollover index if larger than 5GB to a new index incremented by
one.
    options:
      name: metricbeat
      conditions:
        max_size: 5gb
      extra_settings:
        index.number_of_shards: 1
        index.number_of_replicas: 1
      timeout_override:
      continue_if_exception: False
      disable_action: False

  2:
    action: delete_indices
    description: >-
          Delete indices when their cumulative size is larger than 50GB.
      options:
        ignore_empty_list: True
      filters:
      - filtertype: pattern
        kind: prefix
        value: metricbeat-
      - filtertype: space
        disk_space: 50
        use_age: true
        source: creation_date
```