Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Pyry Laasonen

Monitorointiratkaisu tuotantopalvelimille

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

19.02.2019

Tekijä Otsikko Sivumäärä Aika	Pyry Laasonen Monitorointiratkaisu tuotantopalvelimille 36 sivua 19.02.2019
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Tietoverkot
Ohjaajat	Lehtori Jukka Louhelainen Tuotekehityspäällikkö Hannes Haataja

Tämän opinnäytetyön tarkoituksena oli etsiä uusi tuotantopalvelimien monitorointiratkaisu Pandia Oy:lle. Yrityksessä koettiin, että nykyinen monitorointityökalu on liian hankalakäyttöinen. Uuden ratkaisun haluttiin olevan helppokäyttöisempi ja vähentävän turhia hälytyksiä.

Työ toteutettiin perehtymällä yleisesti monitoroinnin perusteisiin. Tämän jälkeen kartoitettiin saatavilla olevia kaupallisia ja ei-kaupallisia monitorointiratkaisuja, joista otettiin vertailuun kymmenen mahdollisesti tuotantopalvelimille soveltuvaa ratkaisua. Näistä valittiin kaksi käytännön kokeiluun, jossa ne asennettiin omille palvelimilleen ja konfiguroitiin monitoroimaan testipalvelinta. Näin selvitettiin käytännönläheisesti niiden soveltuvuutta yrityksen tarpeisiin. Kokeilun perusteella valittiin paremmin soveltuva työkalu otettavaksi käyttöön tuotantopalvelimille.

Työn tuloksena löydettiin toimiva monitorointiratkaisu, joka on helppokäyttöisempi kuin nykyinen. Uutta ei kuitenkaan vielä otettu käyttöön, mutta käyttöönotolle on kuitenkin valmis pohja.

Avainsanat	Monitorointi, Opsview, Sensu

Author	Pyry Laasonen
Title	Monitoring solution for production servers
Number of Pages	36 pages
Date	19 February 2019
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Networks
Instructors	Jukka Louhelainen, Lecturer Hannes Haataja, Product Development Manager

The purpose of this thesis was to find a new production server monitoring solution for Pandia Oy. Company felt that the current monitoring solution was too cumbersome to use. The aim of this thesis was to find a new monitoring solution that is easier to use and to reduce unnecessary alarms.

The work was carried out by studying the basics of monitoring. After that available commercial and non-commercial monitoring solutions were mapped. From those, ten possible solutions for production servers were picked for comparison. Two of the monitoring solutions in the comparison were selected for practical testing, where they were installed on their own servers and configured to monitor available test server. In this way, their suitability as a new monitoring solution for the company was tested in more practical manner. Based on this practical test the more suitable solution was chosen to be deployed on production servers.

As a result, solution was found that is easier to use than the current solution. However, the new monitoring solution has not been deployed yet but there is ready-made base for future deployment.

Keywords	Monitoring, Opsview, Sensu

Sisällys

Lyhenteet

SaaS	Software as a Service. Ohjelmiston tarjoamista palveluna.
SLA	Palvelutasosopimus. Asiakkaan ja palveluntarjoajan välinen sopimus, jossa määritellään palvelulle tietyt vaatimustasot.
JMX	Java Management Extensions on standardi Java-pohjaisen sovelluksen monitorointiin ja hallintaan.
API	Application programming interface on ohjelmointirajapinta, jolla eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja keskenään.
AMI	Amazon Machine Image. Virtuaalikonetyyppi, jota käytetään virtuaalikoneen luomiseen Amazon-palvelussa.

1 Johdanto

Opinnäytetyö tehdään Pandia Oy:n toimeksiannosta. Tämän opinnäytetyön tarkoitus on tarjota Pandialle uusi monitorointiratkaisu tuotantopalvelimille vertailemalla eri ratkaisuja ja korvata nykyinen valvontaratkaisu Zabbix.

Pandia kehittää SaaS-pohjaisia ohjelmistotuotteita kiinteistöalalla toimiville yrityksille. SaaS, eli Software as a Service, tarkoittaa ohjelmiston tarjoamista palveluna. Tuotteilla yritykset voivat hallinnoida rajoitus-, investointi- ja kiinteistösalkkujaan sekä raportoida ja ennustaa liiketoimintaansa. Käytännössä tämä on vain yksi tuote, mistä asiakkaille avataan ohjelmasta ostetut ominaisuudet.

Tämän työn tavoitteena on, että uusi valvontaratkaisu on kustannustehokas ja vähentää turhia hälytyksiä. Tarpeen on myös helpottaa vikojen etsimistä, ennaltaehkäistä vikatiloja ja näin vähentää työmäärää, joka kuluu ongelmien ratkaisemiseen. Uuden monitorointiratkaisun tarkempia kriteereitä ovat:

- käyttöasteen seuranta
- hälytykset
- statistiikka
- integroitavuus
- helppokäyttöisyys
- ei saa kuormittaa palvelua merkittävästi.

Työ on tarkoitus tehdä vertailemalla kymmentä saatavilla olevaa kaupallista ja ei-kaupallista monitorointiratkaisua. Tämän jälkeen valita näistä kaksi käytännön kokeiluun, jossa ne asennetaan omille palvelimilleen ja konfiguroidaan monitoroimaan käytössä olevaa testipalvelinta. Kokeilun perusteella tuotantopalvelimille otetaan käyttöön näistä kahdesta Pandian ympäristöön paremmin soveltuva työkalu.

2 Teoria

2.1 Monitorointi yleisesti

Informaatioteknologian alalla monitoroinnilla tarkoitetaan sovellus- tai laitteistopohjaista järjestelmää, jolla kerätään tietoa järjestelmistä, ja saadun tiedon analysointia mahdollisten vikojen löytämiseksi. Yleisimpiä monitoroinnin kohteita ovat fyysisten osien, kuten prosessorin käyttöasteen sekä vapaan käyttömuistin valvominen.

Monitoroinnista on nykypäivänä tullut tärkeä osa kaikkia järjestelmiä. Niiden pääasiallinen tarkoitus on varmistaa järjestelmän toimivuus ja havaita ongelmat tai häiriöt mahdollisimman varhain. Viat voivat kehittyä nopeasti tai pitkäaikaisen virheellisen toiminnan takia. Vikojen aikainen havainnointi on tärkeää, koska järjestelmän seisahtuminen voi johtaa huomattaviin taloudellisiin menetyksiin. Jotkut järjestelmät tarjoavat automatisoidun tehtävän suorituksen ongelmien ratkaisemiseksi, jotta voidaan minimoida järjestelmän seisahtumiset.

2.2 Monitoroinnin kohteita

Järjestelmissä on useita tietoja, joita voidaan kerätä. Tässä luvussa käydään läpi opinnäytetyölle olennaisia monitoroinnin kohteita; saatavuus, kapasiteetti ja suorituskyky sekä sovellukset.

2.2.1 Saatavuus

Saatavuuden monitorointi antaa tietoja pääsystä järjestelmiin, palveluihin ja sovelluksiin. Saatavuus lasketaan prosentteina ajasta, jolloin järjestelmä on ollut käynnissä ja ajasta, jolloin saatavuutta on mitattu. [1, s. 241.] Saatavuutta voidaan arvioida esimerkiksi ping-komennolla, jolloin ping lähettää request-paketin laitteelle, johon valvottava laite vastaa omalla reply-paketilla.

Monitoroimalla saatavuutta voidaan todistaa asiakasorganisaatiolle, että järjestelmä toimii oikein ja täyttää sille asetetut palvelutaso- ja saatavuusvaatimukset. Nämä

vaatimukset on määritelty järjestelmän palvelutasosopimuksessa SLA:ssa (Service Level Agreement). Saatavuus on yksi tärkeimmistä mittareista, kun ohjelmistoa tarjotaan asiakkaille palveluna. [1, s. 242.]

2.2.2 Kapasiteetti ja suorituskyky

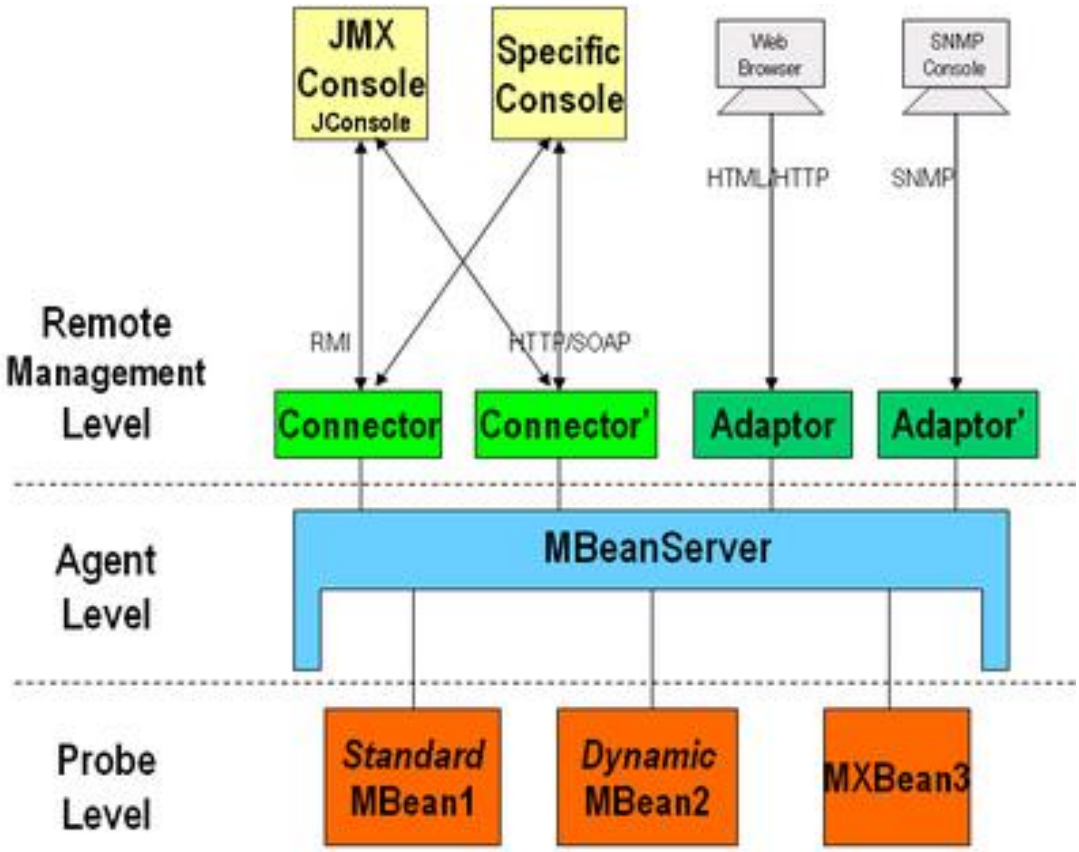
Kapasiteetti ja suorituskyky ovat myös yleisiä tietoja, jotka kerätään järjestelmistä. Kun järjestelmät kasvavat ja käyttävät enemmän resursseja, kerätyt tiedot auttavat ennakoimaan järjestelmän resurssien tarpeita. Näin kerättyjen kapasiteetti- ja suorituskykytietojen perusteella yritykset voivat ottaa paremmin huomioon IT-infrastruktuurin kasvutarpeet. Kerätyt tiedot auttavat myös tunnistamaan, jos järjestelmällä on tarpeettoman paljon tai vähän resursseja käytössä, tai jos jokin muutos aiheuttaa epänormaalia prosessorin kuormitusta tai levytilankäyttöä. Kerätyn tiedon avulla voidaan tehdä asianmukaiset toimenpiteet asian korjaamiseksi. [1, s. 242-243.]

2.2.3 Sovellukset

Sovellusten monitorointi mahdollistetaan usein Application Performance Management (APM) -ohjelmistolla. Tyypillisesti sovellusten monitorointi tarjoaa ajonaikaista metriikkaa sovelluksesta, minkä avulla pystytään arvioimaan sovelluksen suorituskykyä ja saatavuutta.

2.3 JMX

Java Management Extension (JMX) on standardi Java-pohjaisen sovelluksen monitorointiin ja hallintaan. Se mahdollistaa yleisen järjestelmän hallinnan valvoa sovellusta ja luoda ilmoituksia, kun sovellus tarvitsee huomiota sekä muuttaa sovelluksen tilaa ongelmien korjaamiseksi. Valvontaan JMX:ia käytetään useimmiten sovelluksissa, jotta järjestelmää voidaan konfiguroida tai saada tietoa sovelluksen tilasta. JMX rakentuu kolmesta kerroksesta. [2.] Kuva 1 havainnollistaa JMX:n arkkitehtuuria.



Kuva 1. JMX-arkkitehtuuri [3].

Mittauskerros (Probe) mahdollistaa sovelluksien resurssien monitoroinnin ja hallinnan. Se koostuu kehittäjien kirjoittamista sovelluksista, resursseista ja muista hallittavista objekteista. Tässä kerroksessa kehittäjät luovat Java-objekteja nimeltä Managed Beans (MBeans), jotka sisältävät ominaisuuksia ja toimintoja, joita he haluavat paljastaa ulkoisille hallintajärjestelmille. [4.]

Agenttikerros (Agent) on JMX:n ydin. Se koostuu pääasiassa MBean-palvelimesta, jonne rekisteröidään MBeans-objektit. Se toimii välittäjänä mittauskerroksen ja hallintakerroksen välillä. Agentti ohjaa resursseja ja saattaa ne etähallintasovelluksien käyttöön. Agentit ovat usein samalla palvelimella kuin mittauskerros, mutta tämä ei ole pakollista. [4.]

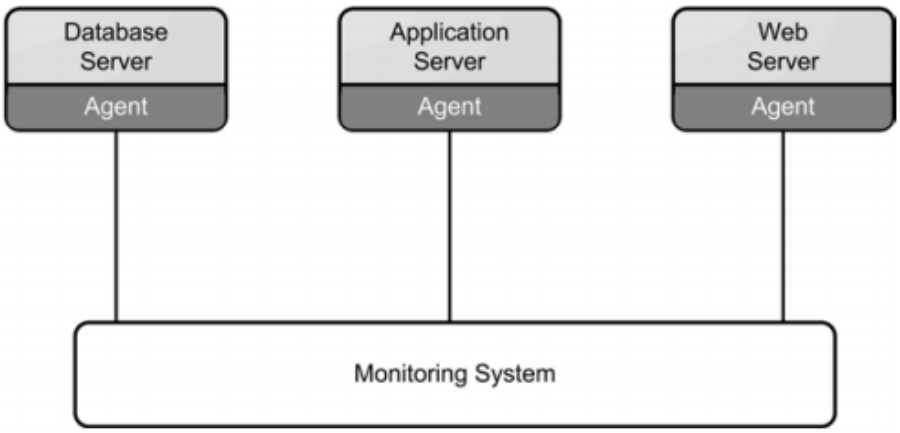
Hallintakerros (Remote Management) koostuu hallintakonsoleista tai muista Java EE -sovelluksista. Hallintasovellus lähettää tai vastaanottaa pyyntöjä agenttikerrokselta. Hallintakerros on usein saatavilla liitännäisenä tai adapterina, joka mahdollistaa hallintakonsolin tukea useita hallintaprotokollia, kuten JMX:sää tai SNMP:tä. [4.]

2.4 Monitorointimenetelmiä

Valvonta tapahtuu yleensä kahdella eri tavalla, joko agentittomasti tai agentillisesti. Hiljattain on kuitenkin esitelty uusia lähestymistapoja. Hybridiratkaisu, joka pyrkii yhdistämään agentillisen ja agentittoman valvonnan parhaat puolet yhteen sekä datavirtamenetelmä, jossa seurantaan liittyviä tietoja kerätään datavirtojen (data streams) kautta. [1, s. 244.]

2.4.1 Agentillinen monitorointi

Useimmiten järjestelmissä käytettävät monitorointiratkaisut ovat agenttipohjaisia, joissa tietoa keräävät järjestelmiin asennetut agenttiohjelmat. Ohjelmat kommunikoivat monitorointijärjestelmän kanssa välittäen keräämäänsä tietoa. Tämänkaltainen ratkaisu mahdollistaa laaja-alaisen tiedon keruun järjestelmästä, sillä agentit ovat verkkotunnuskohtaisia, ja ne on suunniteltu keräämään kaikki mahdolliset tiedot järjestelmästä. [1, s. 244.] Kuva 2 havainnollistaa agenttipohjaista ratkaisua.

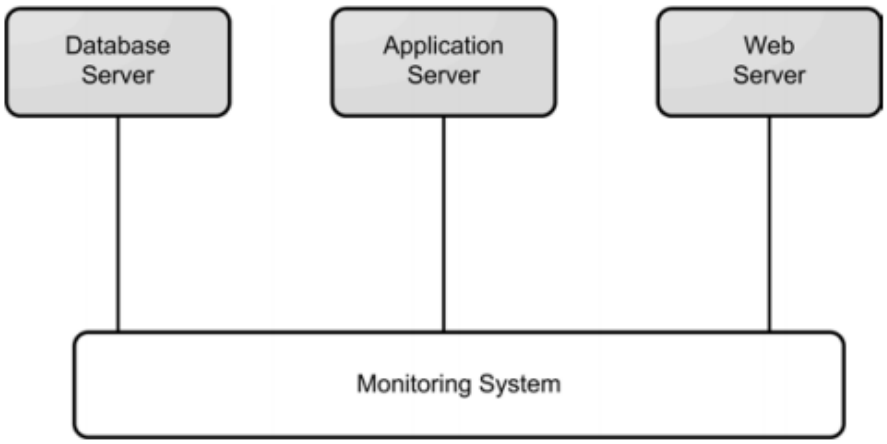


Kuva 2. Agenttipohjainen monitorointiratkaisu [1, s.244].

Agenttipohjainen ratkaisu toisaalta luo rajoituksia skaalautuvuuden kanssa, koska tämänkaltaisen ratkaisun käyttöönotto on hankalaa organisaatiossa, jossa on useita alustoja, järjestelmiä tai sovelluksia. Myös ylläpito ja tekninen tuki ovat haastavampia, esimerkiksi päivitykset on suoritettava jokaiselle palvelimelle, johon agentti on asennettu. Menetelmä vie enemmän resursseja valvottavilta palvelimilta mutta vähemmän verkon kaistanleveyttä. [1, s. 244.]

2.4.2 Agentiton monitorointi

Agentittomissa järjestelmissä valvonta tapahtuu käyttäen järjestelmän sisään rakennettuja valvontateknologioita tai käyttäen liitännäisiä. [1, s. 244.] Kuva 3 havainnollistaa agentittoman järjestelmän toimintaa.

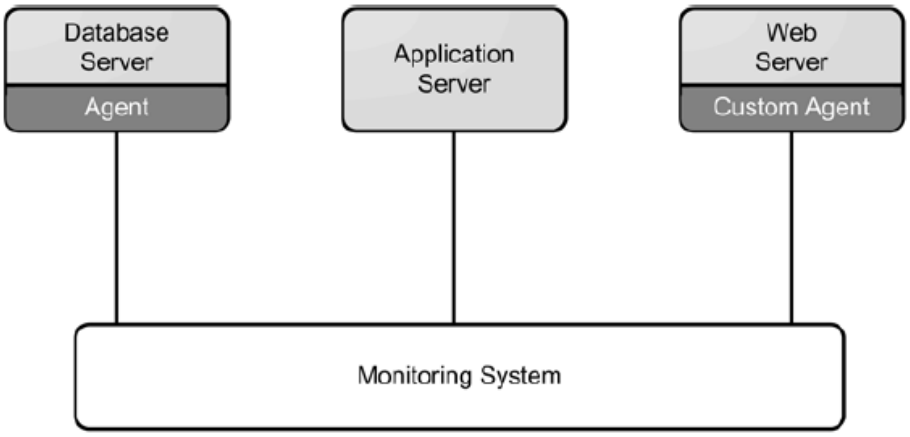


Kuva 3. Agentiton monitorointiratkaisu [1, s.245].

Agentitonta monitorointia pidetään yleisesti helpommin käyttöön otettavana, koska valvottavat palvelimet eivät tarvitse erillistä ohjelmistoa toimiakseen. Tämän takia sitä on myös helpompi ylläpitää, ja se skaalautuu paremmin useille alustoille, järjestelmille ja sovelluksille. Se on kevyt ratkaisu, koska vie vähemmän resursseja valvottavalta palvelimelta kuin agenttipohjainen monitorointi. Agentiton monitorointi kuitenkin luo enemmän verkkoliikennettä, eikä se mahdollista yhtä laaja-alaista valvontaa järjestelmissä kuin agentillinen ratkaisu. [1, s. 244.]

2.4.3 Hybridimonitorointi

Hybridi-menetelmä yhdistää agenttipohjaisen ja agentittoman lähestymistapojen edut. Se antaa valita perinteisten agentillisen ja agentittoman monitorointimenetelmien välillä, mikä mahdollistaa joustavuuden ja skaalautuvuuden valvotuissa järjestelmissä. [1, s. 245.] Kuva 4 havainnollistaa hybridiratkaisun arkkitehtuuria, jossa jokaisessa valvotussa järjestelmässä on sille sopivin monitorointimenetelmä.

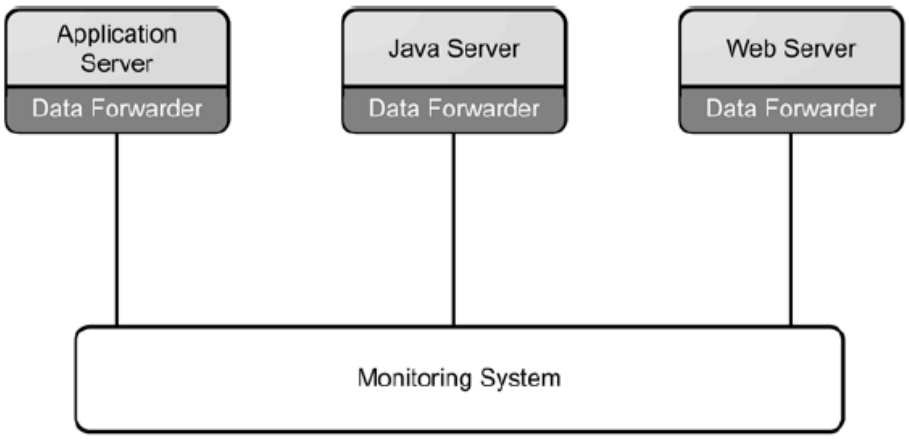


Kuva 4. Hybridipohjaisen monitoroinnin arkkitehtuuri [1, s.245].

Hybridiratkaisussa agenttipohjainen monitorointi voidaan asentaa kriittisiin järjestelmiin, joissa tarvitaan laaja-alaista valvontaa mahdollisten häiriöiden vähentämiseksi. Kevyempää agentitonta monitorointia voidaan taas käyttää järjestelmissä, joissa tarvitsee seurata tavanomaisempia monitorointitietoja, kuten järjestelmän saatavuutta, suorituskykyä ja tilankäyttöä. Tämä ratkaisu kuitenkin luo ylläpidollisia haasteita ja järjestelmien päivittäminen vaikeutuu infrastruktuureissa, joissa käytetään useita eri monitorointimenetelmiä. [1, s. 245.]

2.4.4 Datavirtapohjainen monitorointi

Pilviympäristöjen yleistyessä uusi lähestymistapa kehitettiin mittamaan saatavuutta ja suorituskykyä sovelluksesta palvelun näkökulmasta. [1, s. 246.] Kuva 5 havainnollistaa datavirta-arkkitehtuuria.



Kuva 5. Datavirtapohjaisen monitorointimenetelmän arkkitehtuuri [1, s.246].

Datanlähettäjä asennetaan agenttina palvelimelle, joka lähettää monitorointitietoa virtana valvontajärjestelmälle. Tämä mahdollistaa sovelluksen suorituskyvyn seuraamisen lähes reaaliajassa. Datavirtaratkaisu on samankaltainen kuin agenttipohjainen ratkaisu mutta se on kuitenkin suunniteltu dynaamisemmaksi, ja se keskittyy enemmän koko järjestelmän suorituskyvyn mittaamiseen yksittäisten komponenttien sijaan. [1, s. 246.]

Datavirtapohjainen monitorointiratkaisukaan ei ole ongelmaton. Se tuo samankaltaisia ongelmia kuin agentillinen ratkaisu ylläpidollisesti. Tämänkaltainen ratkaisu vie myös muita enemmän resursseja valvottavilta palvelimilta. [1, s. 246.]

3 Monitorointiratkaisujen vertailu

3.1 Nykyinen monitorointityökalu ja -ympäristö

Pandialla on tällä hetkellä noin 50 palvelinta, jotka toimivat Amazon Web Servicessä eli AWS:ssä. AWS on Amazon.com:n tytäryhtiö, joka tarjoaa SaaS-pohjaisia pilvipalvelualustoja yrityksille ja yksityishenkilöille. Näitä palvelimia monitoroi Zabbix.

Zabbix on ilmainen avoimen lähdekoodin ohjelma, joka on suunniteltu isoihin ympäristöihin. Zabbixin avulla voidaan monitoroida palvelimia, verkkolaitteita, sovelluksia, tieto-

kantoja ja virtuaalikoneita käyttäen agenttipohjaista tai agentitonta menetelmää. Zabbixia kuitenkin pidetään usein vaikeana käyttää ja ylläpitää, kuten Pandiankin tapauksessa.

3.2 Kuinka monitorointiratkaisu valitaan

Uuden monitorointiratkaisun valinta tässä opinnäytetyössä riippuu useasta tekijästä. Tärkein tekijä on kuitenkin työkalun helppokäyttöisyys. Tämä on haaste, sillä monitorointityökalut ovat usein monimutkaisia ja niiden ylläpitämiseen on yleensä koulutetut työntekijät.

Toinen tekijä on kustannukset. Pandialla on tällä hetkellä noin 50 monitoroitavaa palvelinta. Useimpien maksullisten monitorointityökalujen hinnoittelumallit ovat kuukausimaksupohjaisia, ja hinta koskee yleensä yhtä palvelinta. Tämänkaltaiset hinnoittelumallit ovat useimmissa tapauksissa Pandian palvelumallilla kustannustehoton ratkaisu. Hieman suuremmat kustannukset ovat kuitenkin hyväksyttäviä, jos se vaikuttaa vähentävästi työtunteihin.

Kolmas tekijä on ominaisuudet. Uuden monitorointiratkaisun täytyy vähintään pystyä valvomaan käyttöastetta, Java-virtuaalikoneen heap-muistia, prosessorin suorituskykyä, muistia ja palvelimen levytilaa.

Neljäs tekijä ovat hälytykset ja integraatiot. Tavoitteena on vähentää turhia hälytyksiä. Esimerkiksi virhetilanteissa useimmat monitorointityökalut lähettävät hälytyksiä tietyn ajan välein. Uuden ratkaisun tulisi olla muokattavissa niin, että uusintahälytyksiä lähetetään vain tarvittaessa. Tulevaisuudessa kaikki kerätty tieto halutaan mahdollisesti myös yhdistää muunkin tiedon kanssa, joka ei ole mahdollista monitorointityökaluilla, minkä vuoksi työkalun tulisi tukea integraatiota.

Viimeiset tekijät ovat käyttöönotto ja ylläpito. Nämä kulkevat pitkälti käsi kädessä helppokäyttöisyyden kanssa. Käyttöönoton tulee olla yksinkertaista, koska se täytyy tehdä kaikille käytössä oleville palvelimille. Toiveena olisi, että käyttöönoton jälkeen ratkaisu

vaatisi mahdollisimman vähän ylläpitoa. Agentillisessa ratkaisussa täytyy ottaa huomioon, että se pitää asentaa kaikille tuotantopalvelimille. Lisäksi mahdollisista päivityksistä on huolehdittava.

3.3 Vertailu

Markkinoilla on saatavilla paljon kaupallisia ja ei-kaupallisia työkaluja monitorointiin. Työhön valittiin kymmen vertailtavaa työkalua. Nämä 10 työkalua valittiin niin, että ne täyttäisivät mahdollisimman hyvin johdannossa määritellyt vähimmäiskriteerit. Kuva 6 sisältää taulukon, jossa on vertailuun valitut monitorointityökalut.

Työkalu	Lisenssi	Valvontamenetelmä	Käyttöasteen-seuranta	Hälytykset	Integraatiot	Statistiikka	Muuta
Nagios	Avoin lähdekoodi, omisteinen ohjelmisto	Agenttipohjainen ja agentiton	Kyllä	Sähköposti, SMS, mukautettu	Kyllä	Kyllä	Paljon liitännäisiä, laaja yhteisö
Icinga 2	Avoin lähdekoodi	Agenttipohjainen ja agentiton	Kyllä	Sähköposti, SMS, mukautettu	Kyllä	Kyllä	Tukee Nagios liitännäisiä
Check_MK	Avoin lähdekoodi, omisteinen ohjelmisto	Agenttipohjainen ja agentiton	Kyllä	Sähköposti, SMS, mukautettu	Kyllä	Kyllä	Tukee Nagios liitännäisiä
Opsview	Omisteinen ohjelmisto	Agenttipohjainen ja agentiton	Kyllä	Sähköposti, SMS, Slack	Kyllä	Kyllä	Opspacks, tukee Nagios liitännäisiä
Zenoss	Avoin lähdekoodi, omisteinen ohjelmisto	Agentiton	Kyllä	Sähköposti, SMS	Kyllä	Kyllä	Zenpacks
Anturis	Omisteinen ohjelmisto	Agenttipohjainen ja agentiton	Kyllä	Sähköposti, SMS, Puhelu	Kyllä	Kyllä	Useita sisäänrakennettuja sovellusvalvonta teknologioita
Sensu	Avoin lähdekoodi, omisteinen ohjelmisto	Agenttipohjainen	Kyllä	Sähköposti, SMS, API	Kyllä	Kyllä	Konfigurointi tiedostojen automatisointi Chefin ja Puppetin kautta
Pandora FMS	Avoin lähdekoodi, omisteinen ohjelmisto	Agenttipohjainen	Kyllä	Sähköposti, SMS, API	Kyllä	Kyllä	Tukee Nagios liitännäisiä
Datadog	Omisteinen ohjelmisto	Datavirta	Kyllä	Sähköposti, SMS, API	Kyllä	Kyllä	Tukee useita pilvialustoja
OpenNMS	Avoin lähdekoodi	Datavirta	Kyllä	Sähköposti, SMS, mukautettu	Kyllä	Kyllä	Kehitystilastot

Kuva 6. Valvontaratkaisuja.

Vertailussa käyn läpi yleisesti valitsemiani monitorointityökaluja ja niiden ominaisuuksia sekä kustannuksia.

Vertailun perusteella valitsin kaksi monitorointityökalua käytännön kokeiluun, jossa selvitin niiden helppokäyttöisyyttä ja soveltuvuutta Pandian ympäristöön käytännönläheisesti. Näistä kahdesta valitaan Pandialle paremmin soveltuva ratkaisu otettavaksi käyttöön.

3.3.1 Nagios

Nagios on yksi tunnetuimpia avoimen lähdekoodin työkaluja, jolla voi valvoa infrastruktuuria, järjestelmiä, sovelluksia, palveluita ja liiketoiminnan prosesseja. Nagioksesta on kaksi eri versiota; ilmainen avoimen lähdekoodin versio, Nagios Core, ja maksullinen versio, Nagios XI. Nagios Core on supistettu versio Nagios XI:stä. Core-versio mahdollistaa laaja-alaisen monitoroinnin ja hälytykset, mutta raportoinnin ja käyttöliittymän ominaisuuksia on rajoitettu. [6.]

Nagios XI:ssä on paranneltu käyttöliittymä, joka sisältää interaktiiviset hallintapaneelit sekä valvottavien palvelimien yleiskatsauksen. Nagios XI tarjoaa myös parannellun raportointimoduulin, kuten SLA-raportoinnin, sekä mahdollistaa konfiguroinnin käyttöliittymän kautta. Nagios XI:n kustannukset kohoavat kuitenkin useisiin tuhansiin vuodessa. [6.]

Nagioksen asennus on helppoa, mutta konfigurointitiedostojen hallinnointi on haastavaa ja käytön aloittamisessa on jyrkkä oppimiskäyrä. Nagioksella on kuitenkin suuri aktiivinen tukiyhteisö, joka kehittää uusia laajennuksia jatkuvasti. Näillä laajennuksilla voidaan ratkoa joitain Nagioksen puutteita, kuten konfigurointitiedostojen hallinnointiongelmia. Valmistajan ilmoittamia laitteistovaatimuksia ovat

- 2 GB keskusmuistia
- 40 GB kovalevytilaa
- CentOS- tai RHEL-käyttöjärjestelmä. [6]

3.3.2 Icinga 2

Icinga on vuonna 2009 julkaistu ilmainen avoimen lähdekoodin monitorointityökalu, joka pohjautui alun perin Nagiokseen. Se on sittemmin kirjoitettu uudelleen uuteen Icinga 2 -versioon. Sen tarkoituksena on paikata Nagioksen puutteita sekä lisätä uusia ominaisuuksia. Icingassa on Nagioksen ilmaisversioon verrattuna nykyaikaisempi käyttöliittymä ja kattava API (Application programming interface) eli ohjelmointirajapinta, joka mahdollistaa eri ohjelmien tiedonvaihdon keskenään. Nämä ohjelmointirajapinnat ovat yksi tapa mahdollistaa integraatiot. [7.]

Icingan dokumentaatio on kuitenkin vaikeaselkoinen, koska se olettaa lukijalla olevan paljon pohjatietoa monitorointiarkkitehtuurista. Icingaa myös pidetään Nagioksen tapaan vaikeana käyttää, eikä Icingalla ole niin laajaa tukiyhteisöä ongelmien ratkomiseksi.

Valmistaja ei ole ilmoittanut työkalun laitteistovaatimuksia, koska ne riippuvat liian paljon valvottavasta ympäristöstä, kuten valvottavien laitteiden määrästä sekä kuinka monta eri tarkistusta jokaisella laitteella suoritetaan ja kuinka usein. [7.]

3.3.3 Check_MK

Check_MK alkoi alun perin Nagios-työkalun laajennuksena, jonka tarkoitus on skaalautua paremmin. Check_MK:sta on kolme versiota: ilmainen avoimen lähdekoodin Raw Edition ja maksulliset Enterprise ja Managed Services Edition. Ilmainen Raw Edition on muiden monitorointityökalujen tapaan supistettu ominaisuuksiltaan ja siinä käytetään vielä Nagioksen kerneliä eli käyttöjärjestelmän ydintä. Raw Editionilla pystyy luomaan kuvaajia sekä luomaan mukautettuja hälytyksiä. [8.]

Maksullisissa versioissa on Check_MK:n oma kernel Check_MK Microcore, mikä mahdollistaa lyhyemmät tarkistuksien välit ja kuormittaa vähemmän palvelinta. Maksullisilla versioilla pystyy luomaan raportteja ja konfiguroimaan useimmat asiat graafisen käyttöliittymän kautta. Kattavin Managed Services Edition mahdollistaa työkalun paremman mukautettavuuden omaan käyttöön. Maksullisten versioiden vuosikustannukset ovat 600 eurosta reiluun tuhanteen euroon. [8.]

Check_MK on suunnattu isoille monitorointiympäristöille. Dokumentaatio näyttää kattavalta, ja se tukee Nagioksen liitännäisiä. Valmistajan ilmoittamia laitteistovaatimuksia ovat

- 4 GB keskusmuistia
- 80 GB kovalevytilaa
- Ubuntu-, Debian-, Red Hat Enterprise Linux-, SuSE- tai CentOS-käyttöjärjestelmä. [8]

3.3.4 Opsview

Opsview alkoi vuonna 2003 kokoelmana laajennuksia Nagios Coreen. Sittemmin siitä on kehittynyt oma monitorointityökalu, joka keskittyy fyysisten, virtuaalisten ja pilvipohjaisten järjestelmien monitorointiin. Opsview on ilmainen 25 monitoroitavaan järjestelmään asti. Tämän jälkeen yhden monitoroitavan laitteen kustannukset kuukaudessa ovat eurosta kolmeen euroon riippuen monitoroitavien laitteiden määrästä ja kuinka hyvän käyttäjätuen tarvitsee. [9.]

Opsview pystyy tarpeiden mukaan monitoroimaan joko agentittomasti tai agentillisesti. Työkalu sisältää Ops-paketteja. Nämä ovat liitännäisiä, jotka on luotu toimimaan mahdollisimman pienellä konfiguraatiolla. Opsviewin konfigurointi onnistuu yleensä pelkän käyttöliittymän avulla. Opsview mahdollistaa hyvin muokkautettavat hälytykset ja integraatiot Ops-pakettien tai API:n avulla. Opsviewiin on myös rakennettu kattavat raportointiominaisuudet. [9.]

Opsviewilla on selkeä ja laajahko dokumentaatio työkalun käyttämisestä. Työkalulla pystyy myös käyttämään Nagios-standardiliitännäisiä. Valmistajan ilmoittamia laitteistovaatimuksia ovat

- 6 GB keskusmuistia
- 20 GB kovalevytilaa
- Ubuntu-, Debian-, Red Hat Enterprise Linux-, CentOS- tai Oracle Linux-käyttöjärjestelmä. [9.]

3.3.5 Zenoss

Zenoksesta on kolme versiota: ilmainen Zenoss Core ja maksulliset Zenoss On Premises ja Zenoss Cloud. Ilmainen Zenoss Core on rajoitettu versio maksullisesta versiosta. Tämä versio sisältää vain tavalliset monitorointimahdollisuudet, hälytykset ja rajoitetun raporttien luomisen. Zenoss sisältää myös Zen-paketteja. Nämä ovat liitännäisiä, jotka on suunniteltu toimimaan agentittomasti ja mahdollisimman pienellä konfiguroinnilla. Useimmat näistä kuitenkin on saatavilla vain maksullisiin versioihin. [10.]

Maksullisilla Zenoss-versioilla on mahdollista monitoroida reaaliajassa datavirta-menetelmän avulla. Ne myös mahdollistavat SLA-raportoinnin sekä virhetilan analysoinnin. [10.] Valmistaja ei kuitenkaan suoraan ilmoittanut maksullisen versioiden kustannuksia, mutta vuosikustannukset ovat useita tuhansia euroja.

Zenossin dokumentaatio on kuitenkin vaikeaselkoinen, ja työkalun käyttäminen vaikuttaa vaikealta. Valmistajan ilmoittamia laitteistovaatimuksia ovat

- 2 GB keskusmuistia
- 20 GB kovalevytilaa
- Red Hat Enterprise Linux- tai CentOS-käyttöjärjestelmä. [10]

3.3.6 Datadog

Datadog on suunniteltu pilvipohjaisten palvelimien, tietokantojen, sovelluksien ja palveluiden valvomiseen. Datadog käyttää valvomiseen datavirtapohjaista monitorointimenetelmää. Datadog sisältää monipuoliset integraatiomahdollisuudet työkalun valmiiden integraatioiden sekä kattavan API:n myötä. Datadog sisältää myös valmiit työkalut sovelluksien monitorointiin. [11.]

Työkalussa on panostettu kerätyn tiedon graafiseen näyttämiseen ja sen analysointiin. Datadogilla on käyttöön ja asennukseen kattava dokumentaatio, mutta työkalun kustannukset ovat kuitenkin todella korkeat. Valmistaja ei ole myöskään ilmoittanut työkalun laitteistovaatimuksia.

3.3.7 OpenNMS

OpenNMS on ilmainen avoimen lähdekoodin monitorointityökalu, joka käyttää datavirtapohjaista menetelmää monitorointiin. Sitä kehittää ja tukee yhteisö käyttäjiä ja kehittäjiä sekä OpenNMS Group, joka tarjoaa työkalulle kaupallisia palveluita, koulutusta ja käyttäjätukea. [12.]

OpenNMS sisältää kattavan API:n, jolla mahdollistetaan integraatiot sekä erittäin muokkautettavan käyttöliittymän. OpenNMS:llä on laaja dokumentaatio työkalun käytöstä mutta se kuitenkin vaikuttaa sekavalta ja sitä pidetään vaikeakäyttöisenä, etenkin alussa. Valmistajan ilmoittamia laitteistovaatimuksia ovat

- 2 GB keskusmuistia
- 20 GB kovalevytilaa
- 2 CPU
- Ubuntu-, Debian-, Red Hat Enterprise Linux-, CentOS- tai Windows-käyttöjärjestelmä. [12]

3.3.8 Anturis

Anturis on vuonna 2013 julkaistu hieman vähemmän tunnettu monitorointityökalu. Anturiksella monitorointi onnistuu joko agentillisesti tai agentittomasti [13]. Anturis sisältää kaikki aiemmin määritellyt vähimmäisvaatimukset, joiden pitäisi olla asennuksen jälkeen lähes käyttövalmiita. Anturis ei kuitenkaan tarjoa sen enempää ominaisuuksia, mitä voisi tulevaisuudessa hyödyntää. Anturiksen kustannukset ovat noin euron luokkaa per palvelin kuukaudessa.

Anturiksesta ei ole myöskään paljon käyttäjäkokemuksia, ja dokumentaatio on suppeahko, eikä valmistaja ole ilmoittanut työkalulle laitteistovaatimuksia.

3.3.9 Sensu

Sensu on agenttipohjainen monitorointityökalu. Sensusta on kaksi versiota. Ilmainen avoimen lähdekoodin versio Sensu Core ja maksullinen versio Sensu Enterprise. Sensu-työkalu koostuu moduuleista, joten se on hyvin mukautettavissa omiin tarkoituksiin.

Core-versio sisältää monitorointityökalulle tavalliset ominaisuudet sekä kattavan API:n. [14.]

Sensu Enterprise-versio sisältää Coren ominaisuuksien lisäksi käyttöliittymän, jossa on sisäänrakennettu kuvaajien luominen. Se sisältää myös kattavammat integraatiomahdollisuudet sekä käyttäjätuen. [14.]

Sensun työkalun dokumentaatio on selkeä ja kattava, mutta monista Sensun omista liitännäisistä puuttuu dokumentaatio kokonaan. Sensun tarkistukset käyttää kuitenkin samaa standardia kuin Nagioksen, joten Sensun kanssa voidaan käyttää myös Nagioksen liitännäisiä. Valmistajan ilmoittamia laitteistovaatimuksia ovat

- 4 GB keskusmuistia
- 4 GB kovalevytilaa
- Ubuntu-, Debian-, RHEL-, CentOS-, macOS- tai Windows-käyttöjärjestelmä. [14.]

3.3.10 Pandora FMS

Pandora FMS on vuonna 2004 julkaistu monitorointityökalu, joka on suunniteltu joustavaksi. Pandorasta on kaksi versiota: ilmainen avoimen lähdekoodin versio Pandora FMS ja maksullinen versio Pandora FMS Enterprise. Ilmainen versio sisältää kaikki tarvittavat ominaisuudet kuvaajien luomisen sekä hälytykset ja raportit. [15.]

Maksullisessa Enterprise-versiossa on keskitytty enemmän työkalun mukautettavuuden parantamiseen. Käyttöliittymää voi muokata käyttäjäkohtaisesti. Maksullisessa versiossa on myös tuotu mahdollisuus monitoroida palvelimia agentittomasti ja luoda kehittyneempiä SLA-raportteja. Kerättyä tietoakin pystyy tallentamaan pidemmältä ajalta. [15.]

Pandoralla on myös laajahko kirjasto omia liitännäisiä, mutta monet näistä vaativat maksullisen version työkalusta toimiakseen. Valmistajan ilmoittamia laitteistovaatimuksia ovat

- 2 GB keskusmuistia
- 20 GB kovalevytilaa

- Ubuntu-, Debian-, Red Hat Enterprise Linux-, CentOS-, SLES-, OpenSUSE- tai Windows Server-käyttöjärjestelmä. [15.]

3.4 Käytännön kokeiluun valitut ohjelmat

Jokaisella valvontaratkaisulla on omat vahvuutensa ja heikkoutensa. Vertailun ilmaisten monitorointityökalujen ominaisuuksien välillä ei ollut huomattavasti eroja, mutta maksullisten ja ilmaisten työkalujen ominaisuuksien välillä oli eroja. Useimmilla maksullisilla vaihtoehdoilla oli tavanomaisten ominaisuuksien lisäksi sisäänrakennettu raportointityökalu ja kattava kokoelma lähes valmiita tarkistuksia, joiden konfigurointi onnistuu käyttöliittymän kautta. Käyttöliittymät olivat useimmilla käyttäjäystävällisempiä ja ne olivat paremmin mukautettavissa. Kaikki vertailun maksulliset työkalut sisältävät käyttäjätuen.

Eroja oli myös vertailun työkalujen kustannuksissa, dokumentaatioissa ja helppokäyttöisyydessä. Helppokäyttöisyyttä on kuitenkin vaikea arvioida ilman käytännön kokeilua, koska internetissä kirjoitetut artikkelit ja käyttäjäarviot eivät välttämättä anna oikeanlaista kuvaa työkalusta.

Dokumentaatiot olivat useimmilla monitorointityökaluilla selkeät ja kattavat, mikä helpottaa työkalun asennukseen, käyttöön ja ominaisuuksiin perehtymistä. Muutamalla työkalulla dokumentaatio oli kuitenkin joko vaikeaselkoinen tai muuten suppea.

Maksullisten monitorointityökalujen kustannusten erot olivat huomattavia. Kalleimmat ratkaisut olivat kustannuksiltaan useita tuhansia euroja vuodessa, eivätkä useimmat näistä tarjonneet huomattavia etuja halvempiin vaihtoehtoihin verrattuna.

Ensimmäinen valinta käytännön kokeiluun on Sensu, koska se on ilmainen ja hyvin mukautettavissa oleva työkalu. Sensulla on monia yhteisön tekemiä liitännäisiä, joita voidaan hyödyntää ja siihen sopivat myös Nagios-liitännäiset. Sensussa yleensä käytettävä Uchiwa-käyttöliittymä vaikuttaa yksinkertaiselta ja selkeältä. Sensulla on myös selkeä ja kattava dokumentaatio työkalun asennuksesta ja käytöstä. Sensu kuitenkin toimii agenttipohjaisella menetelmällä, mikä saattaa luoda haasteita käyttöönoton kanssa.

Toinen valinta käytännön kokeiluun on maksullinen työkalu Opsview, koska se vaikuttaa kustannustehokkaalta ratkaisulta ja se on mahdollista saada toimimaan agentittomasti. Opsviewillä on selkeä dokumentaatio työkalun käytöstä, ja se sisältää paljon valmiita ratkaisuja kuten Ops-paketteja, jotka vähentävät tarkistusten konfigurointia. Opsviewiin sopivat Sensun tapaan Nagios-liitännäiset, joten molemmissa käytännön kokeiluissa voidaan mahdollisesti hyödyntää samoja liitännäisiä. Opsviewissä voidaan tarkistukset konfiguroida käyttöliittymän kautta. Opsview sisältää myös priorisoidun käyttäjätuen. Kummankaan käytännön kokeiluun valituista työkaluista ei pitäisi kuormittaa palvelua merkittävästi.

4 Käytännön kokeilut

4.1 Sensu Coren asennus

Ennen Sensu Coren asentamista täytyi asentaa sen riippuvuudet Redis ja RabbitMQ. Sensu käyttää tarvittavien tietojen tallentamiseen Redis-tietokantaa, jonne tallentuu esimerkiksi lista valvottavista laitteista sekä tulokset tarkistuksista. Sensu Coren palveluista palvelin ja API tarvitsevat yhteyden samaan Redis-rajapintaan toimiakseen. Redis-tietokannan asennus oli suoraviivaista Sensun dokumentaation avulla.

Sensu-palvelut käyttävät viestibussia (message bus) kommunikointiin valvottavien laitteiden kanssa. Tämän kommunikointitavan mahdollistaa Sensu Transport -kirjasto. Kirjasto mahdollistaa myös vaihtoehtoisten kuljetusratkaisujen käytön oletusohjelma RabbitMQ:n sijaan. Sensu-palvelut tarvitsevat yhteyden samaan määriteltyyn rajapintaan toimiakseen. Sensun tarkistuspyynnöt ja tulokset julkaistaan viestinä Sensu Transportille ja Sensu-palvelut vastaanottavat nämä viestit tilaamalla (subscribing) ne. Ennen RabbitMQ:n asennusta täytyi asentaa Erlang-paketit, koska RabbitMQ-ohjelma on kirjoitettu Erlang-ohjelmointikielellä.

Kun riippuvuudet saatiin onnistuneesti asennettua, voitiin asentaa Sensu Core -paketit. Paketit sisältävät palvelimen, API:n ja asiakasohjelman. Koska Sensu tarvitsi myös käyttöliittymän, asennettiin vielä Sensun dokumentaatiossa käytettävä Uchiwa-käyttöliittymä.

Kun Sensu Core ja sen riippuvuudet on asennettu isäntäpalvelimelle, täytyy Sensu konfiguroida. Sensu ei sisällä valmiita oletusasetuksia, joten mikään Sensun palveluista ei käynnisty ilman oikeanlaista konfigurointia. Sensu pitää konfiguroida kommunikoimaan RabbitMQ:n ja Redisin kanssa. Jotta Sensu voi kommunikoida Redis-tietokannan kanssa, täytyy luoda redis.json-tiedosto polkuun ”/etc/sensu/conf.d/”, johon määritellään kuvan 7 mukaisesti palvelimen IP-osoite ja portti.

```
{
  "redis": {
    "host": "127.0.0.1",
    "port": 6379
  }
}
```

Kuva 7. RabbitMQ:n ja Sensun välisen kommunikaation konfiguraatio.

Tämän jälkeen Sensu konfiguroitiin kommunikoimaan RabbitMQ:n kanssa. Jotta Sensu-palvelut voivat luoda yhteyden RabbitMQ:hun, täytyy ensin luoda RabbitMQ:lle virtuaali-isäntä (vhost) ja käyttäjätiedot. Tämän jälkeen on tehtävä rabbimq.json konfigurointitiedosto polkuun ”etc/sensu/conf.d”, johon määritellään luotu virtuaali-isäntä ja käyttäjätiedot kuvan 8 mukaisesti.

```
{
  "rabbitmq": {
    "host": "localhost",
    "port": 5672,
    "vhost": "/sensu",
    "user": "sensu",
    "pass": "secret"
  }
}
```

Kuva 8. RabbitMQ:n ja Sensun välisen kommunikaation konfiguraatio.

Sensun dokumentaatio oli kuitenkin RabbitMQ:n konfiguroinnin kohdalla puutteellinen, sillä RabbitMQ tarvitsi myös määrittelyn, mitä porttia sen täytyy kuunnella. Tämä voitiin

määritellä RabbitMQ:n omiin tiedostoihin. Tätä varten täytyi luoda rabbimq.config tiedosto polkuun ”etc/rabbitmq/”, koska sitä ei ollut oletuksena luotu. Kuvassa 9 on RabbitMQ määritelty kuuntelemaan TCP-porttia 5672.

```
[
  {rabbit, [
    {tcp_listeners, [5672]}
    ]
  }
].
```

Kuva 9. RabbitMQ:n TCP-portin 5672 kuuntelun määrittely.

Näiden konfiguraatioiden jälkeen Sensu toimii. Jotta tarkistuksien tuloksia voisi lukea helposti, täytyy Sensu ja Uchiwa-käyttöliittymä konfiguroida keskustelemaan keskenään. Tämä tapahtui luomalla uchiwa.json tiedosto polkuun ”/etc/sensu/”, johon määriteltiin Uchiwan ja Sensun IP-osoitteet ja portit, jota ne kuuntelevat. Kuvassa 10 on esimerkkimäärittely Sensun ja Uchiwan välisestä konfiguroinnista.

```
{
  "sensu": [
    {
      "name": "Site 1",
      "host": "127.0.0.1",
      "port": 4567,
      "timeout": 10
    }
  ],
  "uchiwa": {
    "host": "127.0.0.1",
    "port": 4567,
    "timeout" 10
  }
}
```

Kuva 10. Uchiwan ja Sensun välinen konfiguraatio.

Kun kaikkien osien konfigurointi saatiin suoritettua, voitiin avata selain ja syöttää osoitekenttään palvelimen julkinen IP-osoite. Tämän jälkeen määriteltiin portin numero, johon

haluttiin ottaa yhteys lisäämällä IP-osoitteen perään kaksoispiste ja portin numero. Näin päästiin Uchiwa-käyttöliittymään käsiksi.

4.2 Sensu Core -testipalvelimen lisäys monitorointiin

Sensu Core -isäntäpalvelimen asennuksen ja konfiguroinnin jälkeen voitiin yhdistää valvottava laite Sensuun. Ensimmäisenä piti asentaa valvottavalle laitteelle Sensu Core -paketit. Ne sisältävät Sensu Client -ohjelman, joka toimii agenttina.

Asennuksen jälkeen täytyi tehdä asiakasmäärittely luomalla client.json tiedostopolkuun ”/etc/sensu/conf.d/”, johon piti vähintään määritellä valvottavan laitteen nimi, osoite ja tilaukset. Kun kaikki asiakaspalvelimen konfiguraatiot oli saatu valmiiksi, asiakas saatiin rekisteröityä Sensun isäntäpalvelimelle. Kuvassa 11 on asiakasmäärittely, jossa on annettu asiakkaan nimeksi testi ja IP-osoitteeksi localhost sekä linux-tilaus.

```
{
  "client": {
    "name": "testi",
    "address": "localhost",
    "subscriptions": [
      "linux"
    ]
  }
}
```

Kuva 11. Asiakkaan määrittely.

Asiakasmäärittelyn jälkeen laitteen täytyi tietää, mitä kuljetusratkaisua sen tulee käyttää. Tätä varten piti luoda transport.json tiedostopolkuun ”/etc/sensu/conf.d/”, johon määriteltiin kuljetusratkaisu sekä yrittääkö se uudelleen yhdistää automaattisesti virhetilanteissa. Kuvassa 12 on määritelty asiakas käyttämään RabbitMQ-kuljetusratkaisua.

```
{
  "transport": {
    "name": "rabbitmq",
    "reconnect_on_error": true
  }
}
```

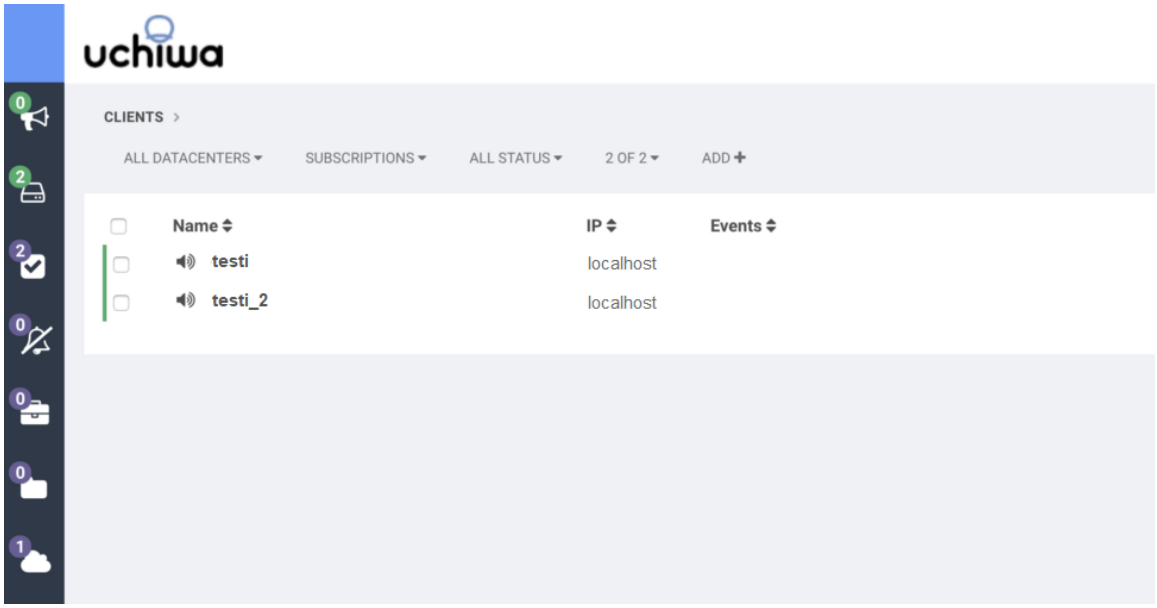
Kuva 12. Kuljetusratkaisun määrittely.

Asiakkaalle täytyi myös antaa tieto, miten yhteys kuljetusratkaisuun luodaan. Kuvassa 13 on esimerkki, miten asiakas isäntäpalvelimeen saa yhteyden. Määrittelyssä on annettu isäntäpalvelimen IP-osoite ja portin numero, jota RabbitMQ kuuntelee, sekä käyttäjätiedot.

```
{
  "rabbitmq":{
    "host": "monitor-test.pandia.fi",
    "port": "5672",
    "vhost": "/sensu",
    "user": "sensu",
    "pass": "secret"
  }
}
```

Kuva 13. Esimerkkimäärittely, miten asiakas saa yhteyden RabbitMQ:hon.

Kun uuden asiakkaan konfigurointi oli valmis, voitiin Uchiwa-käyttöliittymästä käydä tarkistamassa, että uusi palvelin on saatu onnistuneesti monitoroitavaksi. Kuvassa 14 on Uchiwa-käyttöliittymä, jossa on uusi testipalvelin lisätty monitoroitavaksi.



Kuva 14. Uchiwa-käyttöliittymä.

Tämän jälkeen asiakkaalle voitiin lisätä tarkistuksia. Sensulla on paljon Sensu-yhteisön tekemiä avoimen lähdekoodin liitännäisiä, jotka mahdollistavat erilaisia tarkistuksia. Pääsääntöisesti tarkistuksiin määritellään, mitä tarkistusliitännäistä käytetään, millä palvelimilla se suoritetaan ja kuinka usein tarkistus tehdään. Melkein kaikissa tapauksissa täytyy myös määritellä varoitus ja kriittiset kynnysarvot etenkin, jos kyseessä on suorituskyvyn tai kapasiteetin seuranta. Nämä arvot ilmoitetaan yleensä prosentteina mutta ne voivat olla myös lukuarvoja. Kun toinen kynnysarvoista ylittyy, työkalu luo tapahtuman, jolle voidaan konfiguroida käsittelijä, joka tuottaa esimerkiksi hälytyksen.

Uudet tarkistukset luotiin lisäämällä json-tiedosto polkuun ”/etc/sensu/conf.d” ja lisäämällä tiedostoon tarkistuksen vaatimat määrittelyt. Nämä tiedostot yleisesti nimetään tarkistusta kuvaavalla nimellä, kuten ”cpu_check”. Kuvassa 15 on lisätty prosessorin suorituskyvyn tarkistus linux-tilaajille 120 sekunnin välein. Varoituksen kynnysarvoksi on määritelty 80 % ja kriittiseksi kynnysarvoksi 90 % prosessorin kokonaissuorituskyvystä.

```
{
  "checks": {
    "cpu_check": {
      "handlers": [
        "default",
      ]
      "command": "/opt/sensu/embedded/bin/check-cpu.rb -w 80 -c 90",
      "interval": 120,
      "occurences" 5,
      "subscribers": [
        "linux"
      ]
    }
  }
}
```

Kuva 15. Esimerkki prosessorin suorituskyvyn tarkistuksen määrittelystä.

Tarkistuksen määrittelyn jälkeen Sensu-palvelut täytyi käynnistää uudelleen, jotta uudet konfiguraatiot astuivat voimaan. Kuvassa 16 on prosessorin suorituskyvyn tarkistus Uchiwa-käyttöliittymästä nähtynä.

Check	Output	
keepalive	Keepalive sent from client 6 second...	a few s...
cpu_check	CheckCPU TOTAL OK: total=0.0 use...	a minut...

Kuva 16. Onnistunut prosessorin suorituskyvyn tarkistus.

Samaan tapaan saatiin lisättyä myös muistin, levytilan ja saatavuuden tarkistukset. Java-sovelluksen heap -muistin tarkistuksessa jouduttiin kuitenkin tukeutumaan Nagios-yhteisön liitännäisiin, koska Sensu-yhteisön Java-tarkistusliitännäisessä ei ollut minkäänlaista dokumentaatiota. Tämä näyttäisi olevan useimpien Sensu-yhteisön liitännäisten heikkous. Nagios yhteisön liitännäisistä kuitenkin löytyi check_jmx -liitännäinen, joka mahdollistaa Mbeansien ominaisuuksien monitoroinnin.

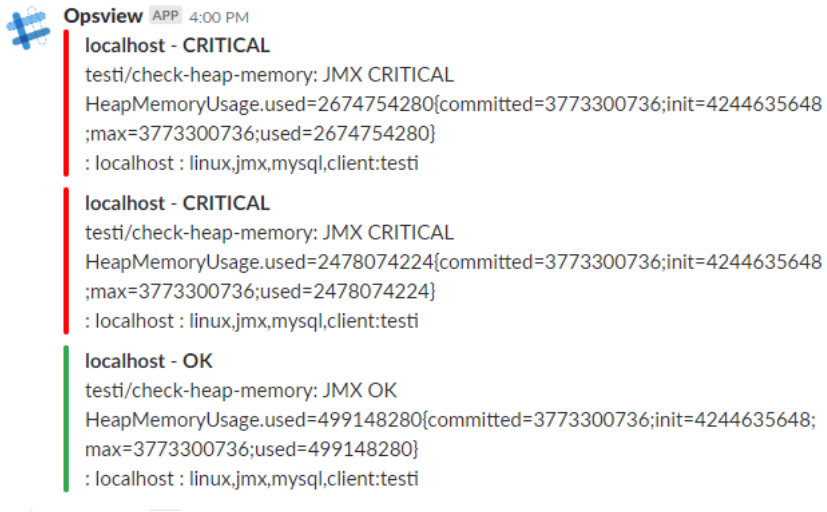
4.3 Sensu Core – Hälytykset

Kun testipalvelin saatiin onnistuneesti monitoroitavaksi, voitiin lisätä hälytyksiä. Sensu-yhteisön kirjastosta saatiin liitännäinen, jolla pystytään lähettämään hälytyksiä Slackiin. Slack on ryhmien ja organisaatioiden sisäiseen viestintään suunnattu pikaviestintäsovellus. Hälytystä varten täytyi konfiguroida käsittelijä Slackille luomalla handler_slack.json tiedostopolkuun ”/etc/sensu/conf.d/”. Tiedostoon täytyi määritellä, mitä liitännäistä käsittelijän tulee käyttää, sekä kynnysarvo, milloin käsittelijä ryhtyy toimenpiteisiin. Tässä tapauksessa täytyi myös määritellä Slack-kanavan nimi, mihin hälytykset julkaistaan, ja webhook. Webhook on http-pyyntötapa, jolla sovellus voi tarjota muille sovelluksille reaaliaikaista tietoa. Kuvassa 17 on käsittelijän määrittely, joka julkaisee hälytyksen Slackissä #opsview-kanavalla, kun tarkistuksen palauttama arvo ylittää kriittisen kynnysarvon.

```
{
  "handlers": {
    "slack": {
      "type": "pipe",
      "command": "/opt/sensu/embedded/bin/handler-slack.rb -j slack",
      "severities": [
        "critical"
      ]
    }
  },
  "slack": {
    "webhook_url": "https://hooks.slack.com/",
    "channel": "#sensu"
  }
}
```

Kuva 17. Käsittelijän konfiguraatio.

Sensulla ei ole mitään käytännöllistä tapaa testata hälytyksiä, joten Slack-hälytyksen toimivuutta testattiin muuttamalla check_jmx-tarkistuksen kynnysarvoja pienemmiksi. Kuvassa 18 on kolme onnistunutta hälytystä Slackin kautta. Hälytyksissä näkyvät tarkistuksien sen hetkinen heap-muistin käyttömäärä ja onko varoitus vai kriittinen kynnysarvo ylittynyt. Viimeisen hälytyksen kohdalla kynnysarvoja nostettiin, jolloin saatiin ilmoitus, että heap-muistin käyttö on palautunut kynnysarvojen alle.



Kuva 18. Onnistuneita hälytyksiä Slackiin.

Sähköpostihälytykset ovat mahdollisia Sensulla, mutta tämä jätettiin tekemättä, koska se olisi ollut ajallisesti kannattamatonta. Sensu olisi tarvinnut sähköpostipalvelimen sekä käsittelijän ja sähköpostipalvelimen konfiguroinnin, jotta sähköpostihälytykset olisivat toimineet. Työn kannalta Slack -hälytykset ovat riittäviä demonstroimaan Sensun hälytyksien toimintoja ja toiminnallisuutta.

4.4 Opsview – Asennus

Opsview-asennus Amazonin pilvipalvelimelle onnistui käyttäen AMI:a eli Amazon Machine Imagea. AMI on valmiiksi konfiguroitu kuvake, jolla luodaan virtuaalikone Amazon Elastic Compute Cloud -instanssiin. Tämä kuvake sisältää valmiiksi käyttöjärjestelmän sekä lisäohjelmat, jotka tarvitaan Opsview-työkaluun. AMI mahdollistaa ohjelman toiminnan heti, kun instanssi on luotu.

Kun instanssi saatiin luotua, voitiin syöttää selaimen osoitekenttään palvelimen julkinen IP-osoite, jolloin päästiin työkalun käyttöliittymään käsiksi. Ensimmäisellä käyttökerralla ohjelma kysyi tuoteavainta, jonka sai rekisteröitymällä palvelun käyttäjäksi Opsviewin sivuilla. Tämän jälkeen käyttäjälle avautui käyttöliittymä ja työkalulle voitiin alkaa lisäämään monitoroitavia laitteita.

4.5 Opsview - testipalvelimen lisäys monitorointiin

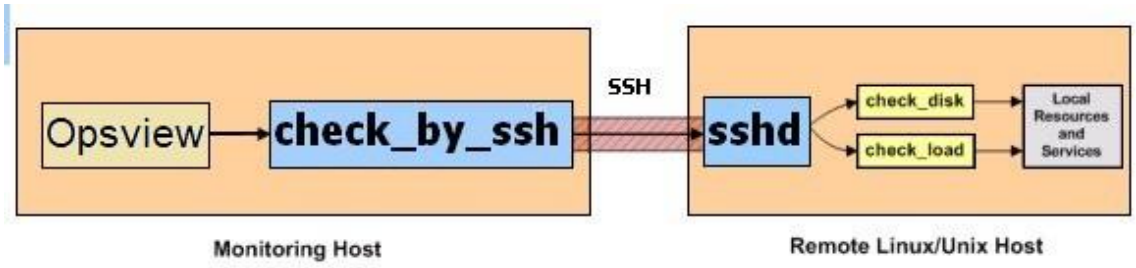
Opsview:llä monitorointi on mahdollista joko agentin kanssa tai agentittomasti. Päätimme toteuttaa kokeilun monitoroinnin agentittomasti. Näin ylläpidon tarve vähenee, ja monitorointi olisi helpompi ottaa käyttöön tuotantopalvelimilla. Agentittomaan monitorointiin on useita eri lähestymistapoja, mutta tässä työssä käytetään SSH:ta. Secure Shell eli SSH on salattuun tietoliikenteeseen tarkoitettu protokolla, jolla voidaan muodostaa kahden laitteen välille turvallinen ja salattu verkkoyhteys.

SSH-yhteys toteutettiin luomalla Opsview-palvelimelle SSH-avain. Tämän jälkeen lisättiin valvottavalle palvelimelle käyttäjä nimeltä opsview. Tämän kotitiedostopolkuun luotiin .ssh-niminen kansio ja lisättiin oikeudet siten, että vain opsview-käyttäjä voi lukea, kirjoittaa ja suorittaa tämän kansion alla olevia tiedostoja. Kansion alle luotiin authorized_keys -tiedosto, johon kopioitiin Opsview-palvelimella luotu julkinen SSH-avain.

Tämän jälkeen SSH-yhteyttä voitiin testata komentorivin kautta. Yhteys ei kuitenkaan toiminut heti, koska palomuuri esti yhteyden muodostamisen. Kun palomuurin tarvittavat portit oli saatu avattua, saatiin yhteys muodostettua testipalvelimelle. SSH-yhteys tuli muodostaa komentorivin kautta ensimmäisen kerran, sillä muuten palvelimen monitorointi ei olisi onnistunut, koska valvottava palvelin ei vielä ollut tunnettujen laitteiden listalla.

SSH-yhteyden muodostamisen jälkeen voitiin kopioida tarkistusliitännäiset testipalvelimelle SSH-protokollan yli käyttäen SCP-komentoa (Secure Copy). Kokeilussa hyödynnettiin pääasiassa Opsviewin sisältämiä tarkistusliitännäisiä. Saatavuuden monitoroinnissa hyödynnettiin kuitenkin Nagios-liitännäistä, koska Opsviewin sisältämä liitännäinen ei antanut dataa sellaisessa muodossa, että siitä saataisiin kuvaaja. Kokeilussa käytettiin myös ajan säästämiseksi samaa check_jmx-liitännäistä kuin Sensu-kokeilussa.

Kun liitännäiset saatiin siirrettyä onnistuneesti testipalvelimelle, voitiin siirtyä määrittelemään tarkistuksia. Niissä hyödynnettiin Opsviewin omaa check_by_ssh-liitännäistä, joka käyttää SSH-yhteyttä tarkistuksiin. Kuva 19 havainnollistaa agentitonta monitorointimenetelmää, joka käyttää tiedonsiirtoon SSH-yhteyttä.

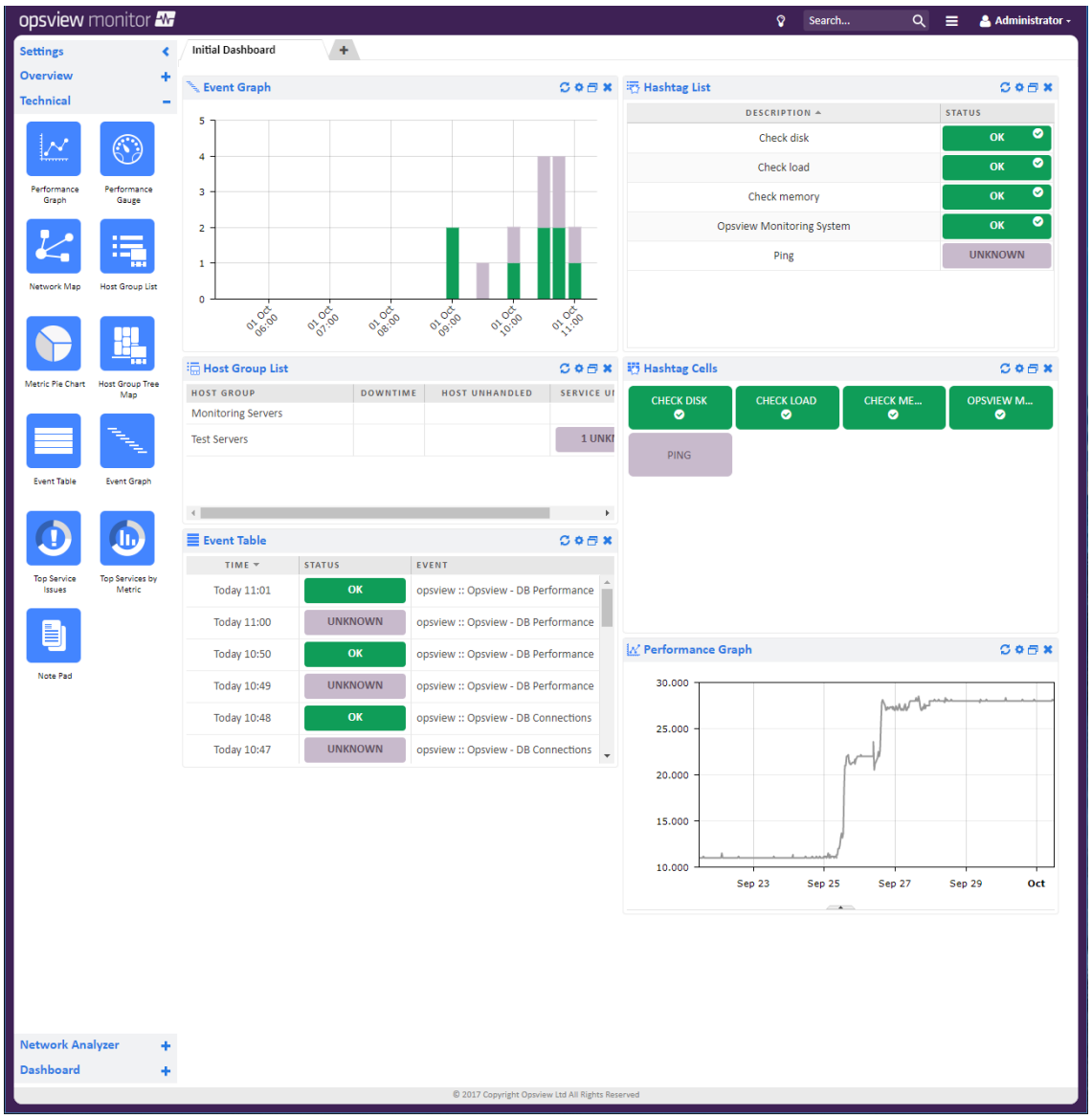


Kuva 19. Agentiton monitorointi SSH-yhteyden avulla [16].

Opsviewin tarkistukset määriteltiin samaan tapaan kuin Sensun, mutta Opsviewillä tämä onnistui käyttöliittymän kautta.

Tarkistuksien määrittelyiden jälkeen voitiin lisätä testipalvelin monitoroitavaksi. Tämä onnistui käyttöliittymän asiakasasetukset-valikosta, jossa voitiin lisätä uusia asiakkaita ja muokata nykyisiä. Määrittelyihin tarvitsi vain lisätä asiakaspalvelimen nimi, IP-osoite ja tarkistukset, jotka haluttiin kyseisellä palvelimella suorittaa.

Kun testipalvelin saatiin onnistuneesti monitoroitavaksi, voitiin käyttöliittymään lisätä erilaisia kuvaajia ja pylväsdiagrammeja tarkistuksien datasta. Näiden avulla voidaan helposti seurata tarkistuksien kehitystä pidemmällä aikavälillä ja estää mahdollisia ongelmatilanteita. Kuvassa 20 on esimerkkejä, miten tarkistuksien tapahtumia ja dataa voidaan havainnollistaa käyttöliittymässä.

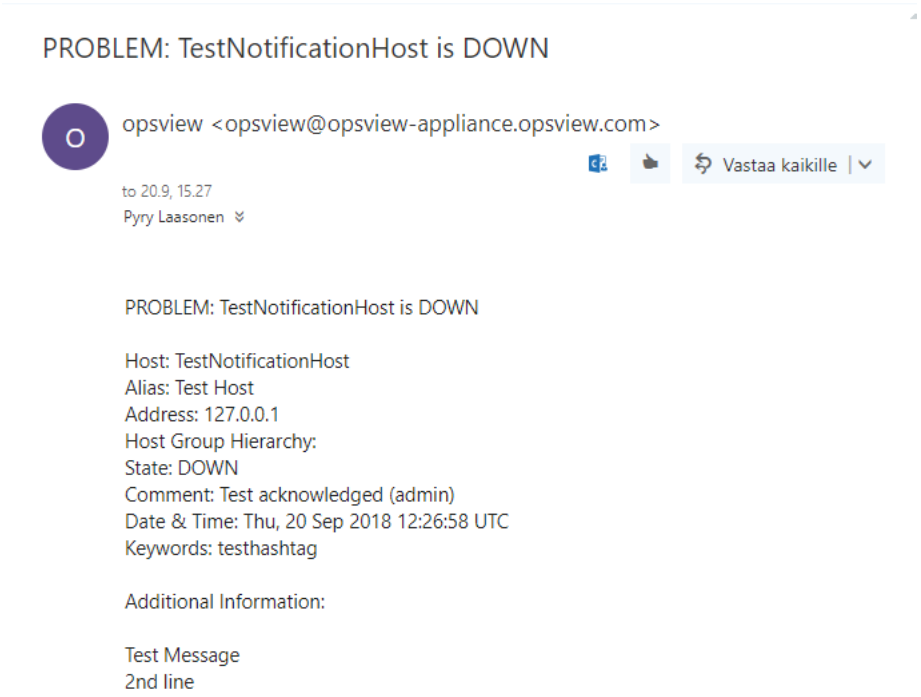


Kuva 20. Esimerkkejä, miten tarkistuksien dataa ja tapahtumia voidaan havainnollistaa käyttöliittymässä.

4.6 Opsview – Hälytykset

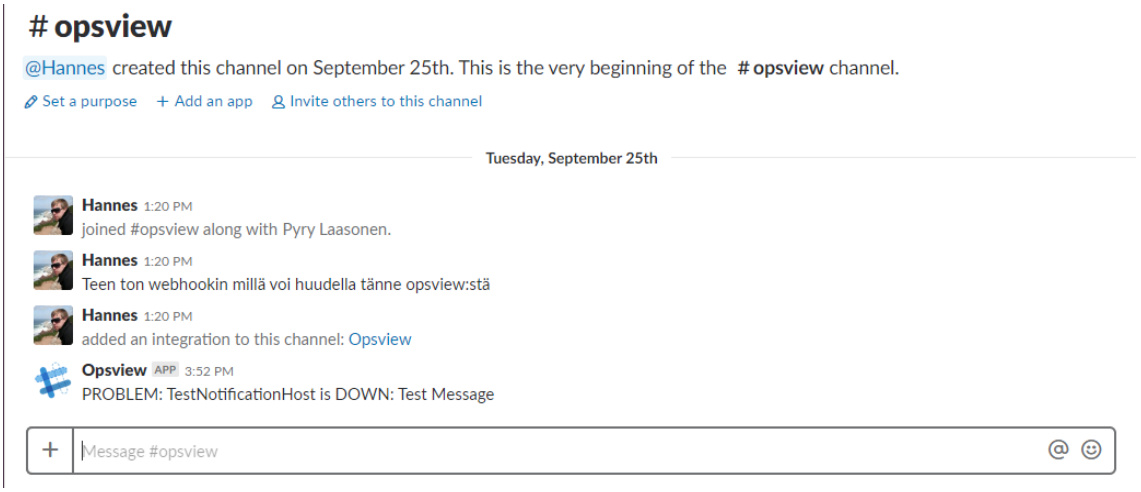
Seuraavaksi lisättiin sähköpostihälytykset. Tämä onnistui lisäämällä Opsviewistä valmiiksi löytyvään sähköpostihälytysmetodiin sähköpostiosoite, mihin hälytys haluttiin. Sen jälkeen täytyi luoda uusi ilmoitusprofiili, johon määriteltiin hälytyksen julkaisutavaksi sähköposti. Ilmoitusprofiiliin pystyi määrittelemään myös, missä tapauksissa hälytys julkaistaan ja kuinka usein. Tämän pystyisi myös määrittelemään palvelinkohtaisesti. Luotua

hälytysmetodia voitiin testata hälytysmetodin alla löytyvällä testipainikkeella. Kuvassa 21 on onnistunut testihälytys sähköpostiin.



Kuva 21. Onnistunut testihälytys sähköpostiin.

Sähköpostihälytyksen jälkeen voitiin konfiguroida hälytyksiä myös Slackin kautta. Hälytysmetodit sisälsivät valmiin Slack-metodin, johon tarvitsi määritellä vain webhook- ja Slack-kanava, mihin hälytykset haluttiin. Tämän jälkeen täytyi muokata hälytysprofiilia julkaisemaan hälytykset myös Slackissä. Kuvassa 22 havainnollistetaan onnistunutta testihälytystä Slackiin.

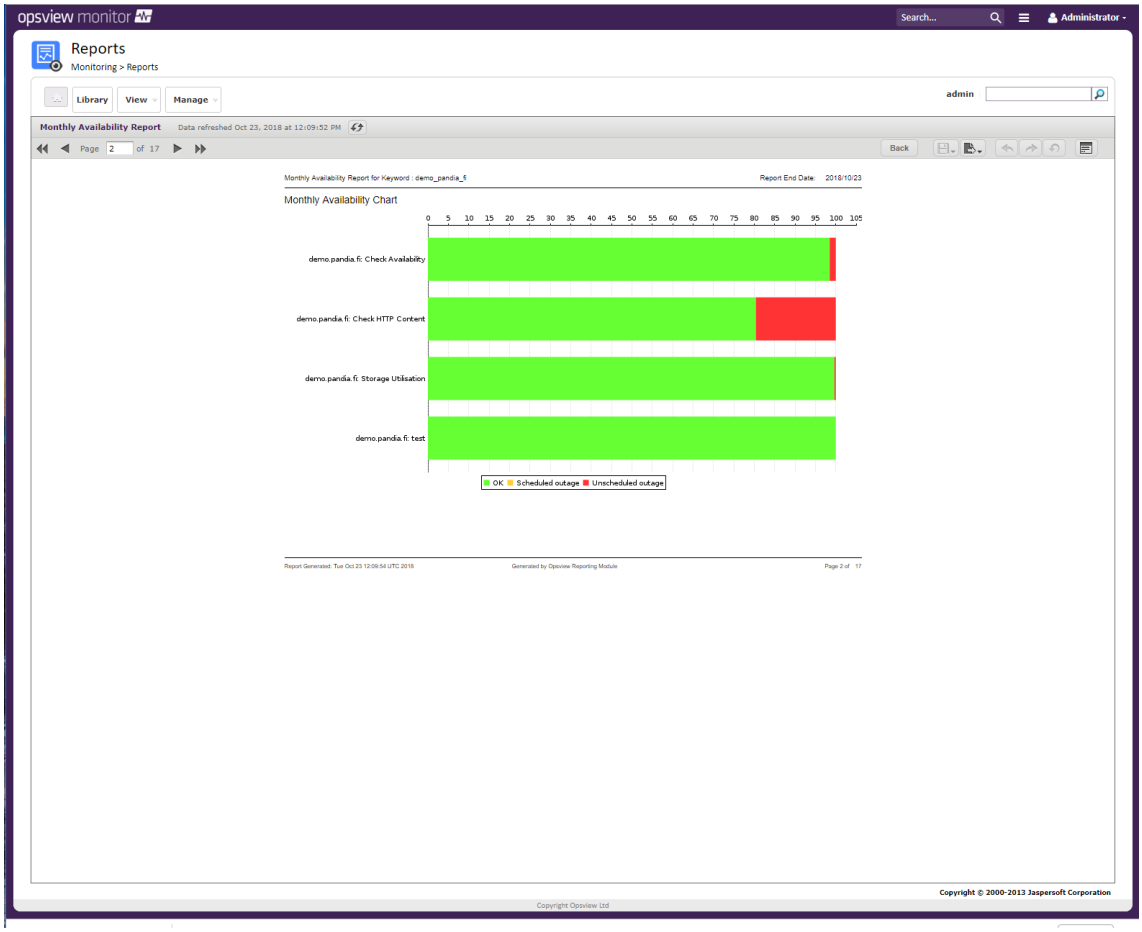


Kuva 22. Onnistunut testihälytys Slackiin.

Opsviewillä hälytykset onnistuvat myös monella muulla eri tavalla, kuten esimerkiksi puhelinsovelluksen kautta, ja tätä voidaan mahdollisesti hyödyntää tulevaisuudessa.

4.7 Opsview – Muita ominaisuuksia

Opsview sisältää myös paljon muita ominaisuuksia, joita voidaan hyödyntää tulevaisuudessa. Se sisältää esimerkiksi raportointimoduulin, joka mahdollistaa raporttien luomisen ja niiden automaattisen generoinnin. Raportointimoduuli sisältää useita valmiita raporttipohjia, kuten suorituskyky-, SLA- ja tapahtumaraporttipohjat. Näitä voidaan generoida päivittäin sekä ladata omalle tietokoneelle eri muodoissa, kuten pdf-, xlsx- ja docx-muodoissa. Kuvassa 23 on generoitu SLA-kuukausiraportti testipalvelimesta.



Kuva 23. Generoitu SLA-kuukausiraportti testipalvelimesta.

Opsview sisältää myös aikaisemmin mainittuja Ops-paketteja, jotka sisältävät valmiita tarkistuksia, joita voidaan mahdollisesti hyödyntää jatkossa.

5 Tulokset

Molemmat käytännön kokeilun monitorointityökalut saatiin onnistuneesti asennettua omille palvelimilleen ja konfiguroitua monitoroimaan testipalvelinta. Käytännön kokeilun aikana havaittiin työkaluissa olevan huomattavan paljon eroja. Suurin osa eroista selittyy varmasti sillä, että Opsview on kaupallinen ja Sensu ilmainen työkalu. Opsview sisältää paljon käyttövalmiita ratkaisuja, kuten hälytykset, raportointimoduulin ja Ops-paketit. Sensu sen sijaan on enemmän työkalu, jonka ympärille voi rakentaa omanlaisen ratkaisun, mikä vaatisi kuitenkin paljon aikaa ja suunnittelua.

Molemmille työkaluille saatiin lisättyä kaikki halutut tarkistukset. Sensun omista tarkistusliitännäisistä kuitenkin puuttui usein dokumentaatio, mikä vaikeutti uusien tarkistusten lisäämistä. Opsviewillä tarkistukset voitiin lisätä vaivattomasti käyttöliittymän kautta mutta saatavuuden tarkistuksessa ei dataa saatu sellaisessa muodossa, että siitä olisi voinut tehdä kuvaajia.

Opsviewillä saatiin Slack- ja sähköpostihälytykset toimimaan vaivattomasti. Opsview käytännössä tarvitsi vain osoitteet, minne hälytykset lähetetään. Opsviewilla on myös mahdollista konfiguroida hälytysten lähetyskriteerit tarkemmin kuin Sensulla. Sensulla saatiin vain Slack-hälytykset toimimaan. Sähköpostihälytykset olisivat tarvinneet toimiakseen sähköpostipalvelimen ja sen konfiguroinnin sekä käsittelijän konfiguroinnin.

Sensulla oli miellyttävämpi käyttöliittymä, mutta se ei suoraan tukenut kuvaajien tekoa ja kaikki muutokset täytyi tehdä komentorivin kautta käyttöliittymän sijaan. Sensu olisi tarvinnut kuvaajien tekoon erillisen lisäohjelman, kuten esimerkiksi Grafanan. Opsviewilla oli mahdollista luoda kuvaajia, kunhan data tuli oikeassa muodossa palvelimelle. Lisäksi etuna oli, että lähes kaikki muutokset voitiin tehdä käyttöliittymän kautta komentorivin sijaan.

Käytännönkokeilun tuloksena tuotantopalvelimien monitorointityökaluksi valittiin Opsview, koska se oli huomattavasti helppokäyttöisempi kuin Sensu ja sisälsi paljon käyttövalmiita ominaisuuksia. Koska Opsview toimi agentittomasti, käyttöönoton pitäisi sujua vaivattomasti. Opsviewin havaittiin kuitenkin kuormittavan jonkin verran isäntäpalvelinta. Kuormituksen skaalautuvuus täytyy selvittää useammalla palvelimella ennen kuin työkalu otetaan käyttöön kaikilla palvelimilla.

6 Yhteenveto

Tämän työn tarkoitus oli etsiä uusi monitorointiratkaisu Pandia Oy:lle nykyisen Zabbix-monitorointityökalun tilalle, koska Zabbixia pidettiin liian vaikeakäyttöisenä. Uuden työkalun haluttiin olevan helppokäyttöinen ja vähentävän turhia hälytyksiä.

Työssä vertailtiin kymmentä eri kaupallista ja ei-kaupallista monitorointityökalua. Näistä valittiin kaksi työkalua käytännön kokeiluun, jotta saatiin selvitettyä käytännönläheisesti,

miten monitorointityökalut soveltuisivat Pandian ympäristöön ja kuinka helppokäyttöisiä ne todellisuudessa ovat.

Käytännön kokeilun perusteella Opsview valittiin otettavaksi käyttöön tuotantopalvelimille. Opsviewille oli helpompi konfiguroida tarkistuksia ja se sisältää enemmän käyttövalmiita ominaisuuksia, kuten esimerkiksi hälytykset, raportoinnin ja kuvaajien teot. Opsviewilla on myös mahdollista toteuttaa monitorointi agentittomasti, mikä yksinkertaistaa ylläpitoa ja käyttöönottoa tuotantopalvelimilla. Opsviewin havaittiin kuitenkin kuormittavan isäntäpalvelinta jonkin verran.

Opsviewia ei kuitenkaan vielä otettu käyttöön. Käyttöönotto tullaan tekemään tämän opinnäytetyön ulkopuolella. Ennen käyttöönottoa tehdään vielä komentosarja, jolla automatisoidaan liitännäisten siirto, käyttäjän luonti ja SSH-yhteyden alustus jokaiselle tuotantopalvelimelle. Näin pystytään nopeuttamaan käyttöönottoa ja vähentämään tähän kuluvaa työmäärää. Myös isäntäpalvelimelle täytyy lisätä resursseja sekä selvittää, kuinka paljon enemmän se käyttää resursseja useammalla monitoroitavalla kohteella.

Lähteet

1 Lukasz, Kufel. 2016. Tools for Distributed Monitoring. De Gruyter Open.

2 Understanding JMX. Verkkoaineisto. <https://docs.oracle.com/cd/E12839_01/web.1111/e13729/understanding.htm#JMXPG114>. Luettu 27.09.2018.

3 Java Management Extensions. Verkkoaineisto. <https://en.wikipedia.org/wiki/Java_Management_Extensions>. Luettu 28.09.2018.

4 Lesson: Overview of the JMX Technology. Verkkoaineisto. <https://docs.oracle.com/javase/tutorial/jmx/overview/architecture.html>. Luettu 28.09.2018.

5 Brattstrom, Morgan & Morreale, Patricia. 2017. Scalable Agentless Cloud Networking Monitoring. International Conference on Cyber Security and Cloud Computing.

6 Nagios Documentation. Verkkoaineisto. <https://www.nagios.org/documentation/>. Luettu 06.09.2018.

7 Icinga 2 Documentation. Verkkoaineisto. <https://icinga.com/docs/>. Luettu 05.09.2018.

8 Check_MK – The official guide. Verkkoaineisto. <https://mathias-kettner.com/cms.html#>. Luettu 05.09.2018.

9 Opsview Knowledge Center. Verkkoaineisto. <https://knowledge.opsview.com/>. Luettu 05.09.2018.

10 Zenoss Core. Verkkoaineisto. <https://www.limswiki.org/index.php/Zenoss_Core>. Luettu 06.09.2018.

11 Datadog Documentation. Verkkoaineisto. <https://docs.datadoghq.com/>. Luettu 06.09.2018.

12 OpenNMS Documentation. Verkkoaineisto. <https://www.opennms.org/en/docs>. Luettu 07.09.2018.

13 Anturis knowledge base. Verkkoaineisto. <https://anturis.userecho.com/knowledge-bases/4-knowledge-base >. Luettu 04.09.2018.

14 Sensu Core. Verkkoaineisto. <https://docs.sensu.io/sensu-core/1.6/ >. Luettu 04.09.2018.

15 Pandora FMS Advantages of Enterprise vs. Open. Verkkoaineisto. <https://pandorafms.com/downloads/advantages-pandorafms-enterprise-vs-open-EN.pdf>. Luettu 10.09.2018.

16 Kumar, Rahul. How to Monitor Remote Linux System with Nagios via SSH. 2014. Verkkoaineisto. <https://tecadmin.net/monitor-remote-linux-system-nagios-via-ssh/>. Luettu 06.10.2018.