UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

MICHEL JUNIOR ISOTON

INTEGRAÇÃO DE FERRAMENTAS PARA A COLETA DE MÉTRICAS EM SERVIDORES LINUX

CAXIAS DO SUL
2019

MICHEL JUNIOR ISOTON

INTEGRAÇÃO DE FERRAMENTAS PARA A COLETA DE MÉTRICAS EM SERVIDORES LINUX

Trabalho de Conclusão de Curso para a obtenção do grau de Bacharel em Sistemas de Informação pela Universidade de Caxias do Sul

Orientadora: Profª. Drª. Maria de Fátima Webber do Prado Lima

CAXIAS DO SUL

2019

MICHEL JUNIOR ISOTON

INTEGRAÇÃO DE FERRAMENTAS PARA A COLETA DE MÉTRICAS EM SERVIDORES LINUX

Trabalho de Conclusão de Curso para obtenção do Grau de Bacharel em Sistemas de Informação da Universidade de Caxias do Sul.

Aprovado em __/__/____

Banca Examinadora

_____
Profª. Drª. Maria de Fátima Webber do Prado Lima
Universidade de Caxias do Sul – UCS

_____
Prof. Giovanni Ely Rocco
Universidade de Caxias do Sul – UCS

_____
Prof. Dr. Ricardo Vargas Dorneles
Universidade de Caxias do Sul – UCS

Dedico este trabalho ao meu pai Federico e minha mãe Ivete, cujo apoio foi fundamental para que fosse possível completar mais essa jornada.

AGRADECIMENTOS

Agradeço primeiramente a meus pais, Federico e Ivete, por todo o apoio e incentivo prestado durante o período de realização deste trabalho e durante minha graduação.

Ao diretor da empresa em que trabalho atualmente, Sr. Aroldo Paulo Bulla, agradeço por ter permitido a redução de meus horários de expediente, possibilitando que eu atenda às orientações e demais compromissos na universidade, além de todo o apoio e positividade.

Agradeço também à minha orientadora, Profª. Drª. Maria de Fátima Webber do Prado Lima, por todas as orientações prestadas e pelo incentivo e colaboração com o desenvolvimento de ideias para o projeto.

À empresa AgênciaNet de Flores da Cunha, também deixo aqui o agradecimento por ter permitido que seu nome fosse utilizado nesta publicação e também por permitir a implementação desta solução dentro do seu ambiente de trabalho e produção.

RESUMO

Este trabalho tem como objetivo encontrar uma solução para a instrumentação (o ato de coletar e monitorar métricas) de servidores Linux em micro e pequenas organizações, onde as soluções já existentes no mercado podem não ser uma boa opção devido a limitações financeiras e/ou de infraestrutura. Para atingir este objetivo, foi feito um estudo da arquitetura de ferramentas de monitoramento existentes, com a finalidade de identificar suas principais características e funcionalidades. A partir deste estudo, foram identificadas quatro funcionalidades básicas necessárias: coleta, armazenamento, visualização e envio de alertas. Para a seleção das ferramentas, foi feito um levantamento de métricas em sistemas computacionais e de critérios de seleção, com o objetivo de selecionar três *softwares* a serem integrados. Os *softwares* selecionados foram o Collectd (*daemon* de coleta de métricas), o InfluxDB (banco de dados de séries temporais) e o Grafana (ferramenta de visualização analítica e envio de alertas). Tais *softwares* foram integrados e a solução foi validada em um contexto real de utilização, onde foi possível constatar que todas as funcionalidades inicialmente planejadas foram desenvolvidas e tiveram um desempenho satisfatório. Durante a validação, foi possível fazer o uso da ferramenta para a coleta, visualização e alertas de métricas de servidores, aumentando a eficiência na utilização de recursos durante a criação de ambientes computacionais e permitindo identificar problemas, tanto na infraestrutura como nas aplicações.

Palavras-chave: Métricas. Linux. Instrumentação. Monitoramento de servidores.

ABSTRACT

This work aims to find a solution for the instrumentation (the act of collecting and monitoring metrics) of Linux servers in micro and small organizations, where existing solutions in the market may not be a good option due to financial limitations and/or infrastructure. To achieve this goal, a study of the architecture of existing monitoring tools was done, in order to identify its main characteristics and functionalities. From this study, four basic functionalities were identified: collection, storage, visualization and alert triggering. For the selection of the tools, a research of metrics in computational systems and selection criteria was made, with the objective of selecting three softwares to be integrated. The selected softwares were Collectd (metrics collection daemon), InfluxDB (time series database) and Grafana (analytical visualization tool and alert triggering). These three softwares were integrated and the solution was validated in a real context of use, where it was possible to verify that all the initially planned functionalities were developed and had a satisfactory performance. During the validation, it was possible to make use of the tool for the collection, visualization and alerts of server metrics, increasing the efficiency of resource utilization during deploy of computational environments and being useful to identify problems, both in infrastructure and applications.

Keywords: Metrics. Linux. Instrumentation. Server Monitoring.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

API	*Application Program Interface*
APT	*Advanced Packaging Tool*
BSD	*Berkeley Software Distribution*
CMIP	*Common Management Information Protocol*
CPU	*Central Processing Unit*
CSV	*Comma Separated Value*
DNF	*Dandified Yum*
ELK	*Elastic, Logstash, Kibana*
ETL	*Extract, Transform, Load*
FIFO	*First In, First Out*
GB	*Gigabyte*
GPL	*General Public License*
GUI	*Graphical User Interface*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
ID	Identificador
I/O	*Input/Output*
IaaS	*Infrastructure as a Service*
IDC	*International Data Corporation*
IOPS	*Input/Output Operations per Second*
JSON	*JavaScript Object Notation*
KVM	*Kernel-based Virtual Machine*
LRU2Q	*Least Recently Used, Two Queue*
MB	*Megabyte*
MIT	*Massachusetts Institute of Technology*
NAND	*Not And*
NAT	*Network Address Translation*
NVMe	*Non-Volatile Memory*
OOM	*Out of Memory*
OSI	*Open Systems Interconnection*
PaaS	*Platform as a Service*

PCIe	*Peripheral Component Interconnect Express*
QEMU	*Quick Emulator*
QL	*Query Language*
RAM	*Random Access Memory*
RDMS	*Relational Database Management System*
RPC	*Remote Procedure Call*
RPM	*Revolutions Per Minute*
RRD	*Round Robin Database*
RSA	Rivest-Shamir-Adleman
SaaS	*Software as a Service*
SATA	*Serial Advanced Technology Attachment*
SMS	*Short Message Service*
SMTP	*Simple Mail Transfer Protocol*
SNMP	*Simple Network Management Protocol*
SSD	*Solid State Drive*
SSH	*Secure Shell*
TI	Tecnologia da Informação
TICK	*Telegraf, InfluxDB, Chronograf, Kapacitor*
TSDB	*Time Series Database*
TSM	*Time Series Merge*
UDP	*User Datagram Protocol*
URL	*Uniform Resource Locator*
WAL	*Write-Ahead Log*
WWW	*World Wide Web*
wSGI	*Web Server Gateway Interface*
XML	*Extensible Markup Language*
YUM	*Yellowdog Updater, Modified*

SUMÁRIO

1. INTRODUÇÃO

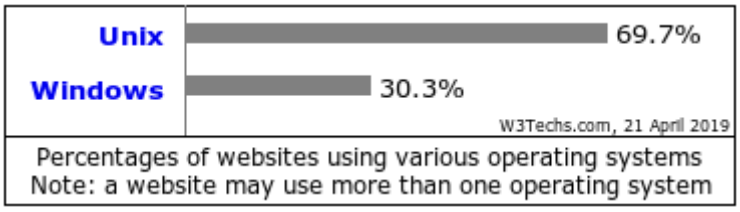
No Brasil, mesmo considerando que os investimentos em Tecnologias da Informação (TI) estão em rápida ascensão e que as projeções indicam um crescimento do mercado em mais de 10% no ano de 2019, as micro, pequenas empresas e microempreendedores individuais ainda têm limitações na implantação e uso de recursos tecnológicos devido a fatores limitadores como custo, mão-de-obra especializada e/ou conhecimento técnico. Estas organizações compõem mais de 99% das empresas brasileiras (DIEESE, 2018), e muitas destas tratam a TI como uma área secundária e não estratégica (PROFISSIONAIS TI, 2014).

Devido às limitações em orçamento para TI nas organizações (CORREIO BRAZILIENSE, 2015), a tendência em migrar aplicações para computação em nuvem vem ocorrendo desde o início da década, e nunca esteve tão em evidência. As micro e pequenas organizações veem nesta migração uma forma de diminuir ou extinguir os gastos em tecnologia *on premise*, que requer aquisição de *hardware*, licenças e criação de procedimentos internos de operação e manutenção além de recursos confiáveis de energia, redundância e rede (GETTI, 2018). De acordo com estudo do IDC, a receita de vendas de produtos de infraestrutura de TI para ambientes de nuvem pública e privada teve um aumento de 47,2% em 2018, chegando a US$ 16.8 bilhões, e pela primeira vez a receita trimestral de fornecedores de vendas de produtos de infraestrutura de TI em ambientes de nuvem ultrapassou a receita de vendas em ambientes de TI tradicionais, representando 50,9% do total das receitas de fornecedores de infraestrutura de TI, contra 43,6% um ano atrás (CONVERGÊNCIA DIGITAL, 2019). Porém, o uso de *cloud computing* não necessariamente é mais barato: a forma diferenciada de cobrança pelo uso de recursos torna os custos mais imprevisíveis, o que faz com que 43% das organizações tenham como maior preocupação o custo deste modelo (SOFTWARE ONE, 2011).

Esta tendência de migração não necessariamente significa o fim do modelo *on premise*: uma estimativa levantada no início do ano de 2019 é de que apenas 20% de toda a infraestrutura das organizações foi levada para ambientes de *cloud computing* (COMPUTERWORLD, 2019). No primeiro momento, as organizações priorizam a migração de sistemas e aplicações simples que já eram compatíveis com este tipo de ambiente, e em um segundo momento analisam os níveis de complexidade de aplicações mais antigas e de maior importância estratégica, nas quais o modelo *on premise* ainda tende a ser empregado. Cerca de 45% das organizações planejam investir mais neste modelo nos próximos 12 meses, ou pelo menos manter os investimentos atuais (SOFTWARE ONE, 2019).

No que tange sistemas operacionais, o uso de opções com código livre é muito amplo: de acordo com pesquisa da W3Techs, considerando o ranking dos 10 milhões de websites mais bem posicionados no ranking Alexa (uma subsidiária da Amazon), 69,7% dos *hosts* fazem uso de um sistema operacional baseado em UNIX, como o Linux (Figura 1).

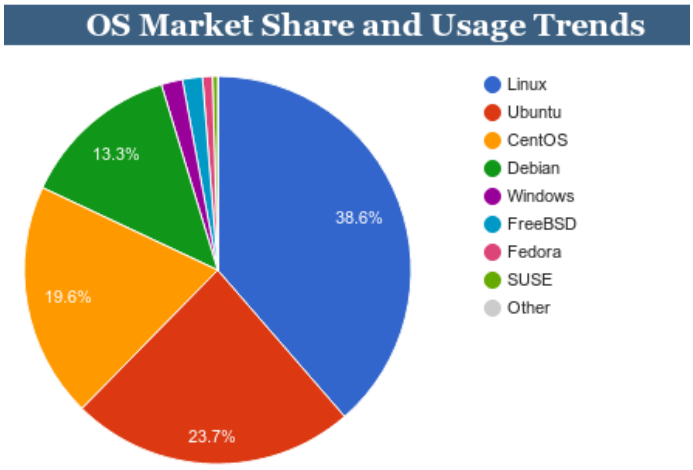
Figura 1 - Ranking W3Tech de sistemas operacionais para servidores WEB



Fonte: https://w3techs.com/technologies/overview/operating_system/all Acesso em: 21 abril 2019

Já em pesquisa da W3 Cook, a qual considera os primeiros 1 milhão de domínios mais bem posicionados do mesmo ranking, concluiu-se que 96,55% dos mesmos servem seu conteúdo a partir de servidores Linux (Figura 2). O fato de se tratar de um sistema operacional gratuito e open-source o torna muito atrativo para micro, pequenas empresas e desenvolvedores que encontram no Linux uma solução econômica, fácil de usar e com uma comunidade convidativa para quaisquer questões de suporte a serem consultadas.

Figura 2 - Ranking W3 Cook de sistemas operacionais para servidores WEB



Fonte: http://www.w3cook.com/os/summary Acesso em: 18 março 2019.

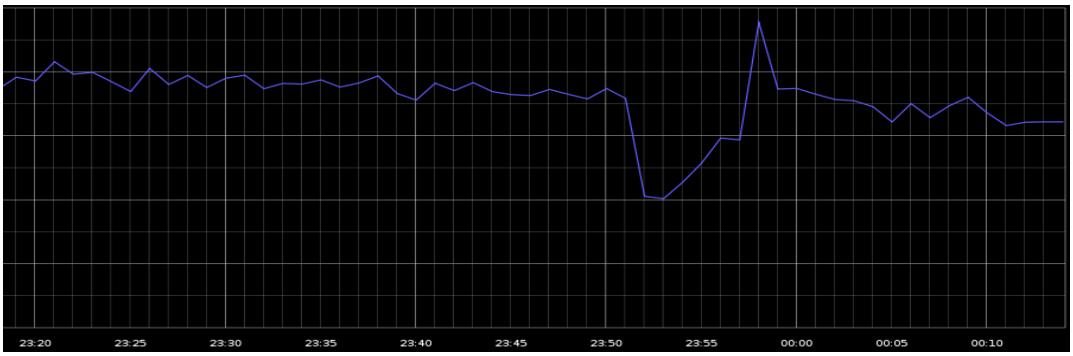
Independente das características do ambiente onde a organização executa suas aplicações, a necessidade de ter acesso e monitorar o estado do serviço e dos recursos computacionais sempre foi algo crítico (TURNBULL, 2016). O correto funcionamento de aplicações exige recursos computacionais suficientes: na falta de tais recursos, aplicações

tendem a travar ou parar de funcionar completamente. Desta forma, o monitoramento de recursos computacionais em servidores tem um papel fundamental na qualidade de serviço e interoperabilidade de aplicações: com uma boa ferramenta, permite-se uma mitigação pró-ativa a um problema ou circunstância que ainda virá a ocorrer, ao invés de uma ação reativa a um problema já ocorrido. Isto pode evitar a interrupção dos serviços de TI, e principalmente a interrupção do funcionamento da organização.

Em 2008, a loja online da *Amazon* ficou fora do ar por duas horas, e estimativas indicam que o prejuízo superou os 30 mil dólares por minuto. Considerando que na época a receita da empresa era de 19 bilhões de dólares e hoje (2019) é de aproximadamente 233 bilhões, pode-se calcular que um possível problema técnico com a mesma duração poderia causar aproximadamente U$ 367.800,00 de prejuízo por minuto se ocorresse hoje em dia, em duas horas, U$ 44.136.000,00 (MOTADATA, 2018).

Em 2013, alguns dos serviços da *Google* ficaram inacessíveis por apenas 5 minutos. Considerando a receita declarada da empresa, estima-se que os prejuízos tenham ultrapassado U$ 454.000,00 para a gigante das buscas. Porém, o impacto negativo também foi sentido por diversos serviços que beneficiam-se do buscador: o tráfego mundial de dados na internet caiu cerca de 40% durante os 5 minutos (TINARI, 2013), como pode ser visto na Figura 3.

Figura 3 - Queda no tráfego mundial de internet durante queda do Google



Fonte: https://engineering.gosquared.com/googles-downtime-40-drop-in-traffic Acesso em: 21 abril 2019

Mais recentemente, em março de 2019, o *Facebook* e serviços relacionados (como os mensageiros instantâneos *Whatsapp* e *Messenger*, e a rede social *Instagram*) enfrentaram problemas técnicos que, de acordo com estimativas, causaram mais de U$ 90.000.000,00 em prejuízos durante 14 horas de instabilidades (BROWN, 2019).

Há de se considerar, ainda, outros gastos provenientes de um incidente. Se o mesmo for grave o suficiente para interromper as operações da organização, a perda de produtividade

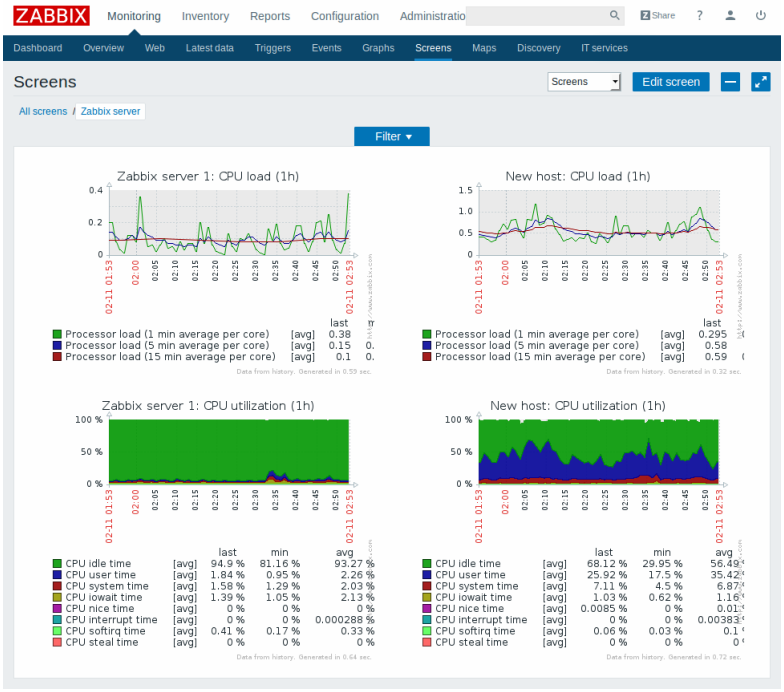
e os custos de recuperação devem ser considerados. Estes últimos compreendem custos de recuperação de dados, ferramentas para reparos e/ou substituições e custos de dados perdidos (ENVIROMON, 2017).

Ainda há gastos que não podem ser mensurados facilmente, como a má reputação da organização, que é diretamente afetada por incidentes externamente conhecidos. Isto é agravado pelo tempo necessário para o restabelecimento dos serviços, e tende a afetar o valor de mercado de uma organização de forma expressiva e rápida visto que mostra claramente que a mesma pode não estar preparada para incidentes inesperados (ENVIROMON, 2017).

Os casos acima citados ocorreram com empresas gigantes da tecnologia, porém os impactos deste tipo de incidente podem ser os mesmos para quaisquer organizações, independente do porte. O monitoramento do status dos serviços poderia garantir uma mitigação quase que imediata, reduzindo possíveis prejuízos.

Para resolver isso, algumas ferramentas surgiram. O Zabbix, na época um projeto interno iniciado por Alexei Vladishev no setor de TI de um banco, foi e até hoje é uma das ferramentas mais populares, principalmente para grandes organizações que possuem vários servidores a serem monitorados, com um grande destaque a coleta de métricas de rede. A Figura 4 ilustra o painel de visualização desta ferramenta.

Figura 4 - Painel de visualização de métricas do Zabbix



Fonte: https://www.zabbix.com/zabbix_web_frontend Acesso em: 17 abril 2019

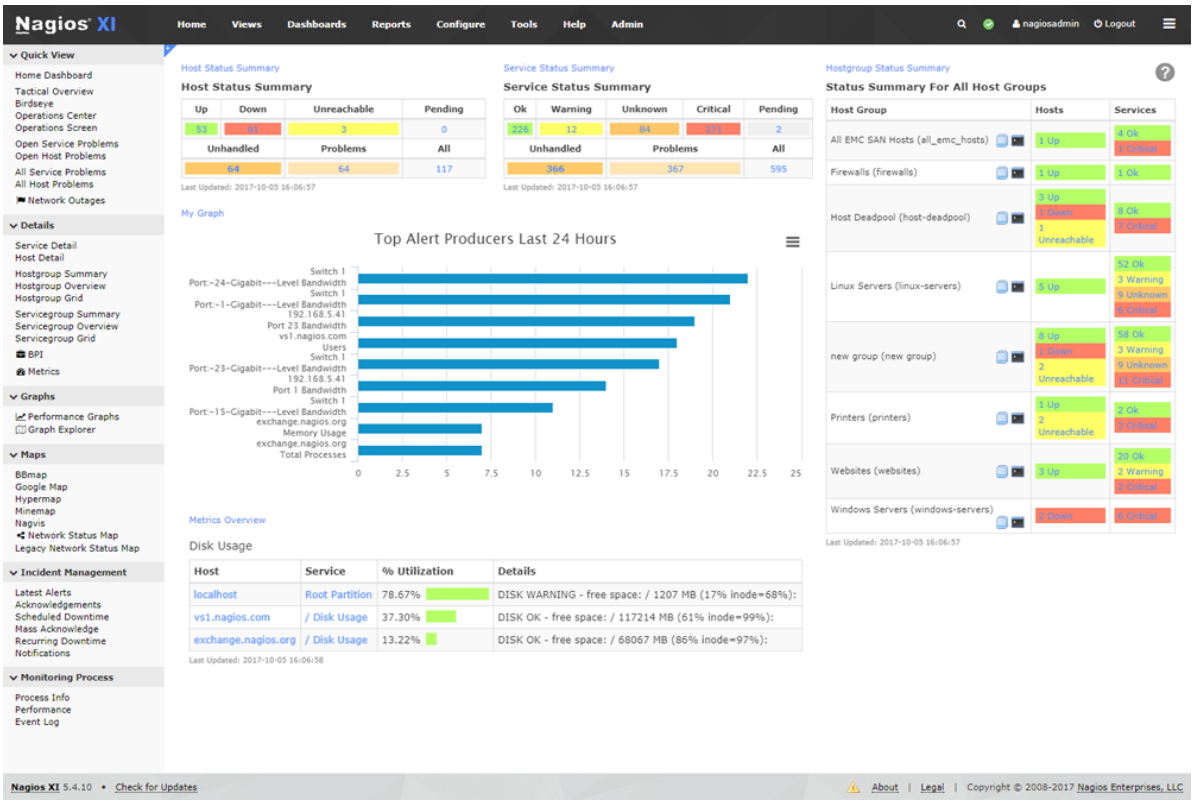
O Zabbix é uma aplicação que consiste em três partes: Zabbix Agent, que é o *software* responsável pela coleta de métricas nos servidores a serem monitorados; Zabbix Proxy, que tem como objetivo aliviar a carga de monitoração atribuída ao Zabbix Server, recebendo dados em seu lugar e encaminhando ao mesmo; e o Zabbix Server, que possui as dependências necessárias para o recebimento, armazenamento (em um RDMS), e visualização em interface WEB.

Existem alguns motivos pelos quais a utilização de uma ferramenta como o Zabbix em micro e pequenas empresas tende a não compensar. A exigência por um ambiente computacional com recursos mínimos relativamente altos é um deles: Embora a documentação oficial do Zabbix 4.0, sua última versão, indique que o requisito mínimo para o Zabbix Server (o componente com requisitos mais exigentes) seja uma máquina com 128 MB de memória física e 256 MB de armazenamento disponível, isto nada mais é que um requisito mínimo. As dependências que precisam executar de forma concorrente no mesmo ambiente possuem requisitos superiores a estes. Faz-se necessária, por exemplo, a instalação de um RDMS para o armazenamento de métricas, de um servidor Apache e do PHP 5.4 com 11 extensões específicas para o funcionamento da interface WEB responsável pela visualização dos dados, e ainda 4 bibliotecas de sistema obrigatórias (ZABBIX, 2018).

Cada uma destas dependências possui requisitos mínimos próprios, que inviabilizam a execução da ferramenta em uma máquina com os requisitos mínimos citados na documentação do Zabbix. O banco de dados tende a exigir uma quantidade considerável de memória devido aos índices, os quais devem ficar comportados na memória RAM (JONES, 2014), e o *tuning* do mesmo, o qual pode resolver esta situação, exige um maior conhecimento em fundamentos de bancos de dados (ORACLE, 2017); o Apache e PHP exigidos já são versões antigas, tendo uma performance até duas vezes inferior a versões mais recentes (SOL, 2019); ainda, há de se considerar o crescimento dos requisitos de sistema devido ao acúmulo de inventório: a quantidade de métricas armazenadas varia conforme a quantidade de máquinas monitoradas, a frequência da coleta de dados, a granularidade da coleta e a quantidade de métricas (ZABBIX, 2018).

Outra ferramenta semelhante ao Zabbix é o Nagios (Figura 5), que atualmente é comercializado como Nagios XI, e seu núcleo continua com código aberto sob licença GPLv2. Seu conceito inicial foi criado por Ethan Galstad em 1996, devido à necessidade em monitorar o estado de servidores Novell, e atualmente mesmo em sua versão gratuita o *software* possui uma ampla gama de funcionalidades (NAGIOS, 2014). Porém, da mesma forma que a ferramenta previamente citada, o Nagios também possui algumas características que são uma desvantagem para micro e pequenas empresas.

Figura 5 - Interface WEB para visualização de métricas do Nagios



Fonte: https://www.nagios.com/products/nagios-xi/ Acesso em: 17 abril 2019

O Nagios é composto tanto de um *software* em servidor, como um *software* (agente) na máquina a ser monitorada. A instalação de ambos exige um conhecimento relativamente amplo na compilação dos códigos-fonte e instalação manual de dependências, além da edição de *templates* de configuração (arquivos de texto). O fato de que há a necessidade da instalação de *plugins* para a coleta de métricas também exige um conhecimento na instalação e configuração dos mesmos, a interface gráfica para a visualização dos dados não favorece uma visão geral de séries temporais, e a falta de interfaces programáticas (APIs) impede a integração com aplicações de terceiros, além de customizações fora do âmbito de *plugins* (SYKES, 2014).

Ao pesquisar *softwares* alternativos que forneçam as mesmas funcionalidades, é possível encontrar referências em uma lista publicada pelo site NetAdminTools, onde constam SolarWinds Network Performance Monitor, PRTG Network Monitor, ManageEngine OpManager, Icinga, OpenNMS.

1.1 PROBLEMA E QUESTÃO DE PESQUISA

A crescente necessidade do uso de ferramentas para a coleta de métricas em ambientes computacionais tem sido suprida por *softwares* como o Zabbix ou o Nagios por quase duas décadas. Ambos desempenham suas funções bem, e possuem características interessantes, porém não necessariamente são uma boa opção para qualquer tipo de organização.

O uso destes *softwares* exige um conhecimento relativamente avançado de instalação e configuração, além de necessitar de recursos computacionais e máquinas extras para o processo de coleta das métricas, as quais tendem a crescer gradativamente. Micro e pequenas organizações precisam de soluções que, embora possam vir a ter menos funcionalidades, atendam a necessidade de coletar e exibir métricas, ao mesmo tempo que são viáveis sem exigir grandes investimentos em infraestrutura e/ou a necessidade de implementação de novas tecnologias.

Segundo SYKES (2014), há excelentes ferramentas disponíveis para cada aspecto do monitoramento de servidores, e ao integrá-las, podemos ter algo realmente funcional e que atenda às exigências sem a complexidade dos *softwares* atuais. Logo, a integração de *softwares* específicos pode ser uma solução viável para resolver o problema.

Para o estabelecimento da questão, é relevante a informação de que, de acordo com dados pesquisados, a grande maioria dos ambientes computacionais utilizados para executar aplicações em meio corporativo estão executando o sistema operacional Linux.

Questão de Pesquisa: Qual a melhor forma de monitorar métricas em servidores Linux em micro e pequenas empresas?

1.2 OBJETIVO GERAL

O objetivo geral deste trabalho foi encontrar uma solução para o monitoramento de métricas em servidores Linux em pequenas organizações, onde a implantação das soluções já existentes no mercado pode não ser uma boa opção devido às limitações de investimentos em pessoal capacitado e/ou infraestrutura.

Para atingir o objetivo deste trabalho, o mesmo foi orientado por quatro objetivos específicos:

- Definir as métricas necessárias para o gerenciamento de servidores;

- Analisar ferramentas de código livre disponíveis;
- Projetar a integração das ferramentas selecionadas;
- Implantar e testar o projeto definido.

1.3 METODOLOGIA

A metodologia adotada neste trabalho foi composta por 7 etapas.

Inicialmente, foi feito um estudo teórico sobre métricas em sistemas computacionais Linux, buscando selecionar as métricas cujo monitoramento é fundamental para a identificação de incidentes ou falhas, além de ter sido feito um levantamento das funcionalidades de ferramentas de monitoramento.

Na segunda etapa foi feito um estudo sobre as ferramentas de monitoramento, bem como foram levantados critérios para seleção de tais ferramentas de acordo com as funções que desempenham e aspectos de compatibilidade de integração, possibilitando identificar qual a melhor alternativa para cada função.

Na terceira etapa, ocorreu o estudo e o desenvolvimento do projeto de integração, onde foram propostos os requisitos funcionais e não funcionais, além da arquitetura e método de validação. Ainda, foram criados protótipos que ilustram o resultado final da integração das aplicações.

Após, na quarta etapa desenvolveu-se as integrações, quando todos os conceitos e tecnologias necessárias foram aplicados durante o processo.

Na quinta etapa foi feito um estudo de caso da implantação e uso da solução proposta na empresa AgênciaNet, de Flores da Cunha/RS. Trata-se de uma empresa de pequeno porte composta por 8 funcionários, cujos principais produtos são aplicações para a WEB, com foco em *websites*. Tal empresa faz o uso de servidores de baixo custo e poder computacional.

Após a implantação, na sexta etapa foram efetuados testes de validação, mensuração de utilização de recursos pela solução proposta, e os resultados foram analisados e validados. Ocorreu ainda uma análise da solução na sétima etapa, como forma de assegurar a utilidade das ferramentas integradas no monitoramento de servidores, e todas as informações levantadas foram utilizadas para concluir o projeto, documentando-se as contribuições, dificuldades e aspectos da utilização.

1.4 ESTRUTURA DO TRABALHO

No capítulo 2 deste trabalho, foi feito um estudo sobre as características e arquiteturas de ferramentas de gerenciamento e monitoramento de elementos de rede, com foco em servidores, além de ter ser feito um levantamento de métricas a serem monitoradas.

No capítulo 3, além da proposta de arquitetura a ser implementada, foi realizado um estudo de critérios para a seleção de ferramentas de coleta, armazenamento e visualização de métricas, além de um levantamento das ferramentas que podem ser integradas para atingir o objetivo de monitorar servidores.

No capítulo 4, foi desenvolvida a proposta de solução para o estudo de caso, com definições referentes à estrutura computacional que veio a ser utilizada, requisitos funcionais e não funcionais, método de validação da solução, além de casos de uso, protótipos e outros aspectos importantes da integração.

No capítulo 5, ocorreu o desenvolvimento da solução proposta. Foram documentados os procedimentos de preparação de ambientes, instalação e parametrização de *softwares*, configuração das ferramentas, e também foi exemplificado o modelo de dados referente às métricas coletadas.

No capítulo 6, todos os casos de testes previamente levantados foram executados, de forma a garantir que as funcionalidades previstas foram implementadas, e foi feita a análise de alguns aspectos importantes da integração das ferramentas, bem como a análise de métricas da mesma com o objetivo de validar a solução.

O capítulo 7 foi utilizado para uma análise da solução proposta no contexto em que fora implantada. Foram feitas análises das informações, explicando o que elas significam e suas utilidades no monitoramento de servidores.

No capítulo 8, foi feito uso de todos os dados coletados durante a etapa de validação da proposta para concluir o trabalho e levantar as conclusões.

2 MONITORAMENTO DE SERVIDORES

Há muitos aspectos a serem considerados ao gerenciar um ambiente computacional. A diversidade de categorias e tipos de componentes, a evolução da tecnologia que faz surgir novos padrões, e o tamanho de tais ambientes que tende a variar conforme o contexto em que estão inseridos são alguns dos fatores mais comuns. Para facilitar o mapeamento das necessidades de gerenciamento, o modelo FCAPS surgiu. No início da década de 1980, o termo foi introduzido nos primeiros rascunhos do que viria a ser a ISO 10040, que define o modelo OSI (*Open Systems Interconnection*, ou Interconexão aberta de sistemas). Naquele tempo, a intenção era definir cinco padrões de protocolo separados, um para cada área funcional (COMER, 2016). Devido à similaridades entre os protocolos, o grupo de trabalho responsável decidiu criar um procotolo único, o qual recebeu o nome de CMIP (*Common Management Information Protocol*, ou Protocolo Comum de Gerenciamento de Informação). Mais tarde, em 1990, o FCAPS foi refinado como recomendação em funções de gerenciamento (COMER, 2016). O Quadro 1 demonstra o modelo.

Quadro 1 - Modelo FCAPS

F	Detecção e correção de falhas
C	Configuração e operação
A	Contabilidade e faturamento
P	Avaliação de desempenho e otimização
S	Garantia de segurança e proteção

Fonte: COMER, 2016.

As necessidades mapeadas aplicam-se a todos os elementos que compõem um ambiente computacional e são passíveis de serem monitorados. Para tanto, faz-se o uso de ferramentas de gerenciamento de rede, as quais COMER (2016) classifica em 12 categorias distintas. Tais categorias estão ilustradas na Figura 6.

Figura 6 - Categorias de ferramentas de gerenciamento de rede

Fonte: Autoria própria.

As ferramentas de gerenciamento são compostas por um *software* gerente e um *software* agente. O agente é instalado no elemento de rede a ser gerenciado, onde desempenha a tarefa de coleta de dados, os quais serão analisados pelo gerente. O gerente é o *software* que recebe e analisa os dados enviados pelos elementos gerenciados, gera ações de controle da rede e elabora gráficos e relatórios, entre outras funções.

As ferramentas de monitoramento de sistemas computacionais têm sido empregadas para o controle de utilização de recursos e de performance de sistemas e redes desde que o conceito surgiu. Estas ferramentas normalmente eram utilizadas por um único administrador e em um único ambiente, porém com o surgimento e desenvolvimento de sistemas distribuídos as ferramentas passaram por uma evolução, adaptando-se à natureza das novas necessidades. De acordo com Fatema et al. (2014, p. 2919, tradução nossa), "Dada a rica arquitetura da *cloud computing*, o monitoramento efetivo requer um conjunto apropriado de ferramentas capazes de monitorar as camadas de IaaS, PaaS e SaaS.".

2.1 FERRAMENTAS DE MONITORAMENTO DE MÉTRICAS

Dentre as ferramentas de monitoramento de métricas, algumas características comuns que se destacam são a capacidade de monitoramento de recursos de sistema, rede e serviços, a natureza *cross-platform* dos agentes, os quais normalmente são compatíveis com ambientes Linux, Windows e em muitos casos Mac, e a licença GPL que permite a execução, adaptação, redistribuição e aperfeiçoamento de tais aplicativos. Ainda, uma funcionalidade comumente

encontrada é a capacidade de gerar alertas via canais de comunicação previamente estabelecidos como *e-mails*, SMS (ou *Short Message Service*) e *callbacks* em HTTP(S), possibilitando o uso de sistemas externos para o envio de avisos customizados. Para uma mitigação rápida a uma possível interrupção de serviço, é comum a configuração do recebimento de alertas em mensageiros instantâneos, como Whatsapp e Telegram (Figura 7).

Figura 7 - Exemplo de alertas enviados pelo Zabbix em uma conversa via Telegram

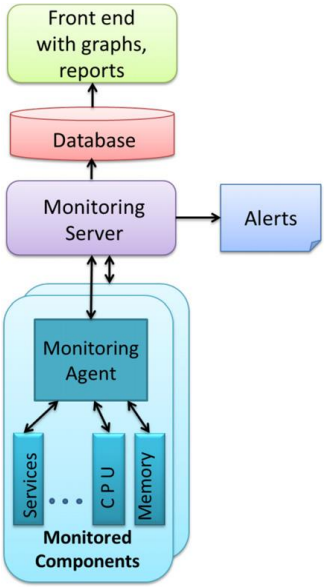
Fonte: https://gabrf.com/ acesso em: 21 abril 2019

De acordo com Fatema et al. (2014, p. 2924), tais ferramentas normalmente possuem algumas limitações, impostas pela natureza generalista dos recursos que possuem e pela tentativa em suportar diferentes tipos de elementos computacionais. É comum, por exemplo:

- A necessidade de configuração manual, devido à ineficiência da funcionalidade de auto-descoberta;
- A necessidade de parametrizar ambos os *softwares*, agente e gerente;

Segundo Fatema et al. (2014, p. 2922), este tipo de ferramenta geralmente utiliza um modelo gerente-agente típico, onde o gerente possui as funcionalidades utilizadas pelo gestor do ambiente: um banco de dados que armazena as informações coletadas, um painel administrativo onde visualiza-se gráficos e métricas, além de relatórios através da manipulação de variáveis, e um componente responsável pela configuração de alertas. A Figura 8 apresenta um modelo desta arquitetura.

Figura 8 - Arquitetura de um software de monitoramento de propósito geral

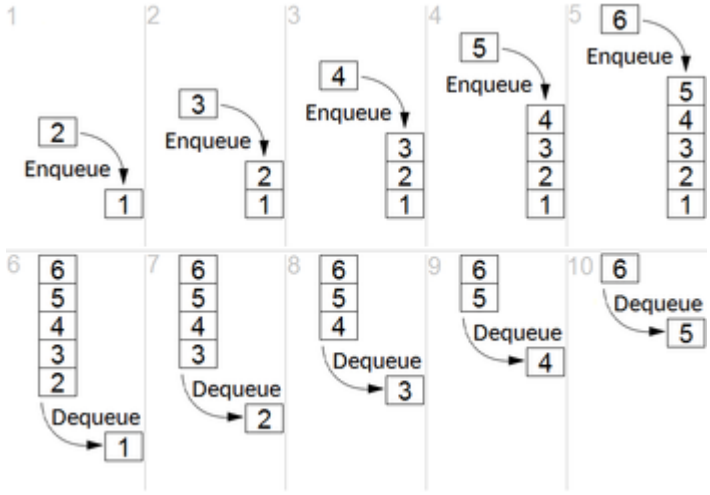


Fonte: FATEMA, et al., 2014, p. 2922.

No que diz respeito às funções desempenhadas, de acordo com Ward e Barker (2014, p. 03) o gerente recebe os dados, os quais normalmente estão ligados a um *timestamp* que representa o exato momento da métrica no ambiente monitorado, e os armazena em um banco de dados. A cada lote de métricas recebido, ocorre a conferência das mesmas junto ao conjunto de regras que definem cada alerta configurado, e ocorre o disparo de alertas se uma ou mais regras analisadas são verdadeiras. Por fim, após o armazenamento em um banco de dados, o componente da ferramenta responsável pela exibição das métricas pode renderizar relatórios e gráficos.

A prática mais comum no que tange o armazenamento de métricas e estatísticas no gerente é o uso de um algoritmo FIFO (*First In, First Out*), ilustrado na Figura 9. Estabelece-se um tamanho máximo que o banco de dados pode atingir, normalmente calculado de acordo com o espaço de armazenamento disponível e com a quantidade de memória RAM que os índices tendem a ocupar, e quando tal espaço é totalmente ocupado, os registros mais antigos são removidos. Isto pode ocorrer de forma automática, visto que alguns bancos de dados possuem tal funcionalidade (seja de forma nativa, ou através de *triggers*), ou através do uso de rotinas.

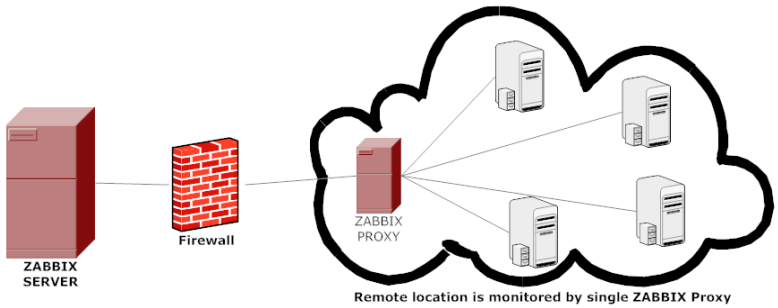
Figura 9 - Algoritmo FIFO em funcionamento



Fonte: https://liag.ft.unicamp.br/conteudos/estrutura-de-dados/ Acesso em: 03/05/2019

Para manter o gerente em funcionamento, faz-se uso de um ambiente compatível, que possua todos os componentes necessários para seu correto funcionamento. Visando eficiência, um único gerente pode receber informações de diversos agentes. Por este motivo, muitas ferramentas passaram a oferecer recursos como ‘gerentes *proxy*’, que diminuem a carga de processamento e memória necessários no gerente, ao receber requisições de agentes e processá-las em máquinas separadas (ACETO et al., 2013, p. 2104). Isto viabiliza o monitoramento de vários agentes simultaneamente, porém aumenta os custos de operação, visto que são necessárias mais máquinas e poder computacional. A Figura 10 demonstra a forma de utilização de um servidor proxy no *software* Zabbix.

Figura 10 - Esquema técnico de uso do Zabbix com um servidor Proxy

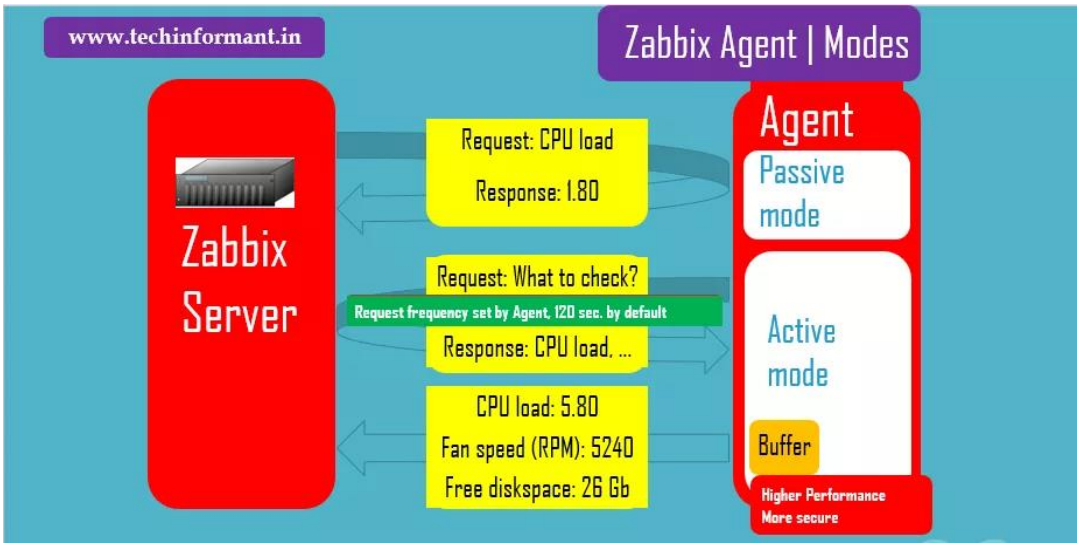


Fonte: https://www.zabbix.com/documentation/3.0/manual/distributed_monitoring/proxies Acesso em: 03/04/2019

O *software* agente coleta as métricas a serem monitoradas através de um serviço do sistema (Daemon). Uma característica importante deste tipo de aplicativo é o baixo consumo

de recursos computacionais (*footprint*) e a capacidade de armazenar as métricas coletadas em um banco de dados próprio para envio posterior ao gerente. O envio dos dados do agente para o gerente ocorre de forma passiva ou ativa (STENICO; LING, 2014, p. 33), como pode ser visto na Figura 11.

Figura 11 - Modos de coleta do Zabbix



Fonte: https://www.techinformant.in/zabbix-agent-installation-on-ubuntu-debian/ Acesso em: 28 março 2019.

A coleta passiva é quando, através de uma requisição, o gerente solicita uma determinada métrica ao agente. O agente acessa sua base de dados e retorna o valor ao gerente, em caso de sucesso. Em caso de falha, um código de erro normalmente é retornado.

A coleta ativa ocorre com uma maior proatividade por parte do agente. Há duas formas básicas: o agente pode possuir um conjunto de métricas pré-estabelecidas a serem coletadas, e envia as mesmas para o gerente em um determinado intervalo, previamente definido. Ou o agente, com uma certa regularidade, requisita ao gerente uma lista de métricas a serem enviadas, armazena a mesma, e realiza a tarefa de coleta e envio em um intervalo definido. A primeira é mais empregada em ambientes controlados, onde as métricas a serem monitoradas já são conhecidas e não mudam com frequência. A segunda é o exato oposto: ambientes dinâmicos onde faz-se necessária a mudança e ajuste de métricas a serem monitoradas.

2.2 MÉTRICAS DE SERVIDORES

Métricas representam as mensurações do uso de recursos ou comportamento que pode ser observado e coletado em um sistema. Ao serem empregadas como fonte de dados em

um sistema de monitoramento, permitem a análise de contexto e obtenção de informações úteis (ELLINGWOOD, 2019).

Pode-se classificar métricas relevantes a serem coletadas em duas categorias: métricas de baixo nível, geralmente provenientes do próprio sistema operacional, e métricas de alto nível, oriundas de aplicações executadas no elemento. Algumas métricas são apresentadas como uma relação entre o valor utilizado e uma capacidade total; outras são uma representação de uma média indicando o estado de ocupação de um componente.

Regularmente, as métricas mais fáceis de coletar são as que são naturalmente expostas pelo sistema operacional e representam a utilização de recursos físicos: dados referentes a espaço em disco, utilização de disco, utilização de CPU, uso de memória e arquivo de paginação (ELLINGWOOD, 2019).

Métricas de alto nível são dependentes das aplicações a que se relacionam. Estatísticas como número de requisições de servidores, ou número de *queries* em um determinado banco de dados, por exemplo, dependem de uma interface disponível para a coleta de dados. A exposição destes dados é denominada instrumentação (ÖZMEN, 2009, p. 910).

Toda captura de dados executada por agentes faz automaticamente a relação entre o dado coletado e o momento da ocorrência. Isto é fundamental, pois situa o dado em relação ao tempo, sendo o tempo a chave principal para operações de agregação e correlação. A agregação permitirá, quando necessário, uma visão geral do sistema em diferentes granularidades; e a correlação será muito útil ao relacionar métricas e reconhecer padrões.

Os tipos de dados a serem monitorados provavelmente mudam à medida que a infraestrutura, e até mesmo as aplicações evoluem. Como elementos de ambientes computacionais tendem a funcionar de forma hierárquica, com camadas cada vez mais complexas sendo construídas umas acima das outras, pode ser útil pensar em métricas disponíveis em diferentes níveis.

2.2.1 CPU

A utilização de CPU, ou de processamento, é uma descrição de como o(s) processador(es) de uma máquina, seja real ou virtual, estão sendo utilizados. Há de se considerar que é comum que sistemas possuam mais de uma unidade central de processamento, e que cada uma pode possuir mais de um núcleo. Não somente isso, cada núcleo pode ou não possuir múltiplos *hyperthreads* (OPSDASH, 2019).

De acordo com Caen e Negus (2018), a forma como estes dados são mensurados diz respeito à quantidade de tempo consumida para a execução de tarefas. Para cada tipo de tarefa, um contador é mantido e incrementado à medida que a tarefa consome tempo de processamento. Pode-se dividir o estado de tarefas em até 10 tipos (Kreeftmeijer, 2018), porém são 7 os mais relevantes:

- *System*: Demonstra tarefas sendo executadas em *kernel mode*;
- *User*: Demonstra tarefas executando em *user mode*, o que inclui aplicações. Há de se considerar que se uma aplicação necessita executar uma operação fora deste escopo (por exemplo, acessar o disco ou fazer uma requisição via rede), a tarefa entra em estado de espera até o *kernel* executar as tarefas necessárias, e retornar;
- *Nice*: Execução de tarefas de usuário em baixa prioridade, normalmente em *background*;
- *I/O Wait*: Execução de escritas e leituras de dados em disco ou rede;
- *Steal*: Quando executado em um ambiente virtualizado, o *hypervisor* pode 'roubar' ciclos de processamento do sistema operacional e 'dar' para outro inquilino, por diversos motivos;
- *Idle*: Quando não há nenhuma tarefa a ser processada, e não há nenhuma operação de I/O;
- *SoftIRQ*: Quando o kernel está em serviço de interrupção de *hardware*.

2.2.2 Memória RAM

O monitoramento de dados de utilização de memória RAM é imprescindível para uma análise de utilização de recursos de um servidor, visto que geralmente, é o primeiro fator limitante ao executar aplicações.

Em sistemas Linux, além da memória RAM física, é habitual a utilização de arquivo de paginação, amplamente conhecido como *swap*. Seu monitoramento também é importante, visto que trata-se de uma partição de tamanho limitado (KAMENOV, 2007).

A utilização de *swap* é relativamente simples: quando um processo requisita ao kernel a alocação de uma determinada quantidade de memória e o sistema esgotou sua memória RAM física, o *kernel* irá buscar os blocos menos frequentemente usados na memória e armazená-los na partição *swap*, comumente presente em disco rígido. Desta forma, libera-se memória física para a alocação (KAMENOV, 2007).

No momento em que os blocos em *swap* se fazem necessários, são movidos de volta à memória física, e um novo bloco (ou conjunto de blocos) menos frequentemente usados são enviados ao *swap*. Este mecanismo permite disponibilizar mais memória às aplicações do que o sistema dispõe fisicamente, o que é denominado 'memória virtual', e é um procedimento totalmente transparente para as aplicações (BOVET; CESATI, 2003, p. 529-531).

Há pontos negativos na utilização de *swap* descontrolada, que potencialmente pode impactar todo o sistema. O mais grave é que a partição de paginação é criada em disco, onde as velocidades de escrita e leitura são muito inferiores às velocidades de memória RAM. Isto faz com que a constante entrada e saída de dados force a CPU a dedicar grande parte do tempo executando tarefas de leitura e gravação em disco. Ainda, isto mantém o disco constantemente ocupado, reduzindo a performance de leitura e escrita de dados legítimos, além de reduzir drasticamente a vida útil do mesmo (SIMS, 2007).

Em sistemas que fazem uso de SSDs (*Solid State Drives*) para o armazenamento persistente (o que é comum em muitos provedores de serviços em cloud computing atualmente), embora o tempo de leitura e gravação seja muito menor, a perda de desempenho ainda é perceptível.

Uma das particularidades do uso de memória RAM em sistemas Linux é o *disk caching*. De uma forma geral, trata-se de um *buffer* transparente para operações em arquivos em disco. Qualquer memória não alocada por aplicações em execução pode ser usada para este cache, e pode ser reclamada a qualquer momento pelo *kernel* para alocação (BOVET; CESATI, 2003, p. 474).

Desta forma, há diferentes métricas de memória que podem ser monitoradas e são relevantes (GITE, 2013):

- *Used*: Total de memória utilizada, que corresponde a *total - free - buffers - cache*;
- *Free*: Total de memória livre;
- *Buffers*: Total de memória usada por *buffers* do kernel;
- *Cache*: Total de memória utilizada para cache;

No que diz respeito a memória *swap*, as métricas são mais simples:

- *Used*: Capacidade utilizada de memória virtual;
- *Free*: Capacidade disponível de memória virtual.

Em algumas distribuições Linux, o próprio sistema operacional possui mecanismos que protegem as aplicações sendo executadas em caso de problemas na memória física. O OOM (*Out of Memory Killer*), por exemplo, é um mecanismo de proteção que em situações de extrema falta de memória, sacrifica processos para recuperar memória baseando-se em um conjunto de regras e heurísticas (KERNEL, 2019).

As decisões feitas por este mecanismo são baseadas em uma pontuação (denominada *oom_score*), aferida a cada processo, e que muda de acordo com a execução das aplicações (CHOUDHURY; PERRETT, 2018, p. 7). Tais heurísticas são pré-definidas, sendo comum que processos fundamentais para as aplicações em execução sejam encerrados, causando interrupções nos serviços das aplicações.

Isto reforça a ideia de que, independente de quanta memória um sistema possua, de mecanismos existentes no sistema operacional, ou das previsões de uso, o monitoramento de métricas referentes à memória é fundamental.

2.2.3 Espaço em disco

O espaço em disco, ou espaço de armazenamento, é uma métrica recorrentemente mensurada por gerentes de infraestrutura. Isto porque aplicações tendem a consumir espaço gradativamente, e o sistema operacional Linux em suas versões '*server'* não apresenta avisos (ou tais avisos não são acessíveis em modo *headless*) quando a máquina atinge sua capacidade total de armazenamento. O comportamento comum é a falha em operações de escrita, o que tende a interromper funcionalidades em execução (INDORE, 2015).

Ainda, o Linux tende a gerar logs de texto documentando alertas, erros, informações e incidentes ocorridos a nível de sistema operacional e de aplicações (normalmente mantidos em /var/log) fazendo uso de servidores de logs como o rsyslog e systemd-journal. Por padrão, não é estabelecido um tamanho limite para este tipo de arquivos, o que frequentemente causa problemas devido ao alto consumo de espaço em disco (MOLINA, 2017).

O espaço em disco total tende a ser um valor constante, sendo alterado apenas em momentos onde efetua-se um upgrade no sistema. A mensuração ocorre por partições de forma individual, sendo que cada partição possui duas métricas:

- Espaço total: Capacidade total da partição;
- Espaço ocupado: Capacidade consumida;

2.2.4 Utilização de disco

Por utilização de disco, entende-se a velocidade de operações concorrentes de entrada e saída de dados executadas em um sistema de arquivos. Trata-se de uma métrica relevante, visto que é comum aplicações executarem tarefas cuja necessidade de leitura e/ou escrita é intensa, o que pode prejudicar o desempenho e responsividade do sistema.

Uma das causas disso é o escalonador de processos de sistemas operacionais Linux, que emprega a técnica de escalonamento circular com prioridades. Nesta prática, processos em estado de I/O são priorizados em relação a processos *CPU bound*, recebendo uma maior parcela de tempo de processamento (*quantum*) como forma de oferecer um melhor tempo de resposta às aplicações interativas (MACHADO; MAIA, 2017) .

Com a popularização de SSDs em servidores, a performance em leitura e escrita de dados em disco teve uma grande evolução. Pelo fato de utilizar memórias NAND para o armazenamento de dados, todo o processo mecânico de operação de discos foi dispensado DALE; LEWIS, 2011). Hoje em dia já é possível encontrar SSDs que atinjam cerca de 100.000 IOPS (input/output operations per second) em interface SATA e mais de 1.000.000 IOPS em interface NVMe e PCIe. Em contraste, os discos rígidos mecânicos de 15.000 RPM atingem, no máximo, 200 IOPS (KOZLOWICZ, 2017).

2.2.5 Utilização de rede

Através de uma ou várias interfaces de rede, o sistema operacional e as aplicações em execução enviam e recebem dados de forma simultânea. No caso de servidores, o monitoramento de fluxo de rede permite identificar, melhor que qualquer outra métrica, os momentos de pico de utilização de aplicações em execução. Isto porque tal fluxo tende a crescer em função do número de utilizadores, considerando que cada utilizador é normalmente uma máquina cliente a ser servida com dados (ARSENAULT, 2017).

As métricas a serem monitoradas são:

- Taxa de entrada: Velocidade em que os dados são recebidos;
- Taxa de saída: Velocidade em que os dados são enviados.

2.3 CONSIDERAÇÕES DO CAPÍTULO

O gerenciamento de servidores é um dos aspectos mais importantes do monitoramento de ambientes computacionais, visto que o correto funcionamento destes elementos depende direta e indiretamente da utilização de recursos computacionais.

Independente do tipo de ambiente ou até mesmo de aplicações a serem executadas, faz-se necessário o gerenciamento como forma de garantir a interoperabilidade e qualidade dos serviços, o que atribui valor à operação.

Para suprir a demanda de monitoramento, algumas ferramentas com o enfoque em coleta e visualização de métricas surgiram. São ferramentas comumente compostas por um *software* agente, o qual coleta e envia dados ao gerente, e o *software* gerente, que analisa, gera ações e elabora gráficos e relatórios das métricas recebidas. Tais métricas tendem a pertencer a um dos dois grupos: métricas de alto nível e métricas de baixo nível.

Métricas de baixo nível indicam o estado de ocupação e/ou utilização de recursos da máquina monitorada como uso de CPU, uso de memória RAM, uso de disco e fluxo de rede.

Métricas de alto nível são provenientes de aplicações em execução no sistema. Normalmente exigem o desenvolvimento de interfaces para a captura de dados, num processo denominado ‘instrumentalização’. É comum a captura de estatísticas referentes a RDMSs e servidores WEB, por exemplo.

O estudo de arquiteturas de ferramentas existentes e da relevância da coleta e análise de métricas, explícito neste capítulo, é o primeiro passo para uma proposta de solução para o problema da coleta de métricas em servidores Linux.

3 ESCOLHA DAS FERRAMENTAS

Após um estudo sobre os componentes que fazem parte do gerenciamento de redes, pode-se concluir que há quatro funções básicas prioritárias a serem supridas: coleta, armazenamento, visualização e alertas. Essas quatro funções básicas podem ser realizadas por três tipos de ferramentas:

- Ferramentas de coleta: *softwares* agentes responsáveis por coletar as métricas nos elementos computacionais a serem monitorados;
- Ferramentas de armazenamento: bancos de dados que possibilitem o armazenamento de dados, de forma que fiquem classificados em função do tempo de recebimento;
- Ferramentas de visualização: *softwares* para a visualização de métricas e composição de gráficos e relatórios, que também permitam a configuração de alertas.

Para atingir o objetivo deste trabalho, neste capítulo foi feito um levantamento de ferramentas disponíveis que se enquadram nos três tipos acima descritos. Foram formulados critérios para cada tipo, de forma que a seleção considerou todos os requisitos exigidos, tanto pelas ferramentas como pelo contexto onde a solução veio a ser aplicada.

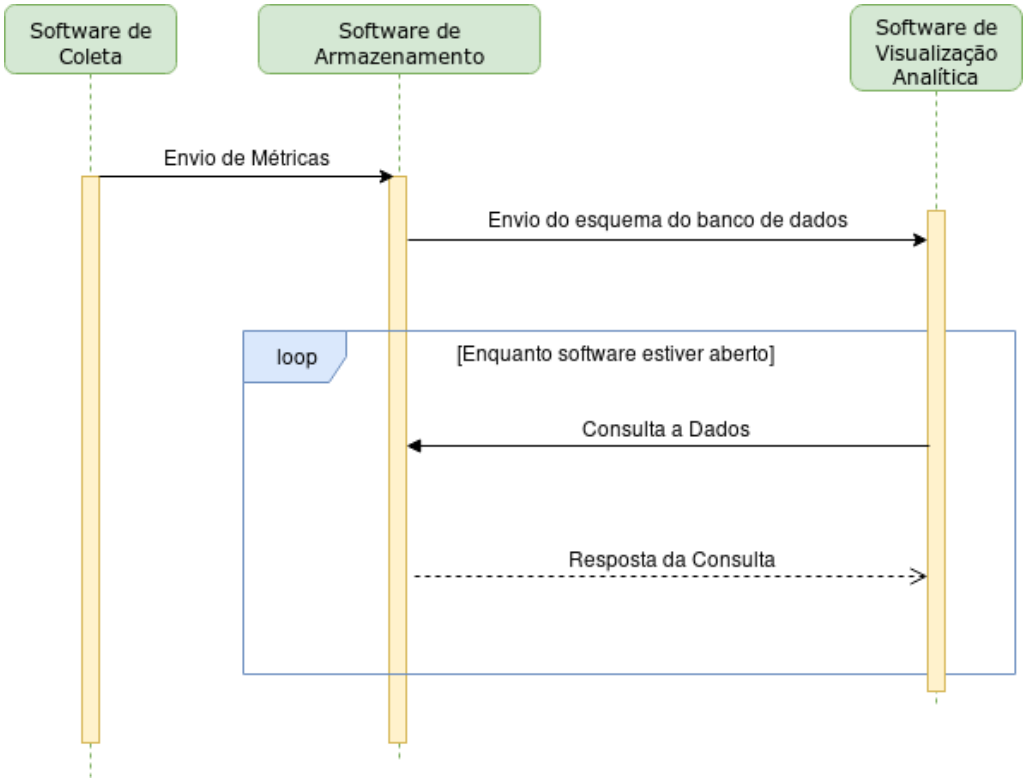
3.1 FINALIDADE DAS FERRAMENTAS SELECIONADAS

Como analisado no capítulo 2, ferramentas de monitoramento de elementos computacionais são geralmente compostas por duas partes: um *software* agente, e um *software* gerente. Existem exceções, como quando o *software* fica na própria máquina monitorada, juntamente com o agente.

Esse trabalho teve como objetivo criar um modelo que possui uma máquina dedicada para a aplicação gerente, a qual recebe as métricas coletadas em uma ou mais máquinas monitoradas para armazenamento e visualização. Porém, ao invés de fazer uso de um *software* único que desempenhe todas as tarefas necessárias para o monitoramento de servidores, o projeto contemplou a integração de um conjunto de ferramentas: foram integradas instâncias de agentes com um banco de dados de séries temporais e uma ferramenta de visualização analítica.

Os agentes (*softwares* de coleta) fazem a coleta de métricas nas máquinas a serem monitoradas; o banco de dado, instalado na máquina dedicada, faz o armazenamento de séries temporais; e a ferramenta de visualização analítica permite, através de componentes gráficos, a visualização das métricas bem como a manipulação das mesmas, e a configuração de alertas a serem enviados. A Figura 12 demonstra, através de um diagrama de sequência, a arquitetura da integração de uma forma geral.

Figura 12 - Diagrama de sequência da integração dos softwares



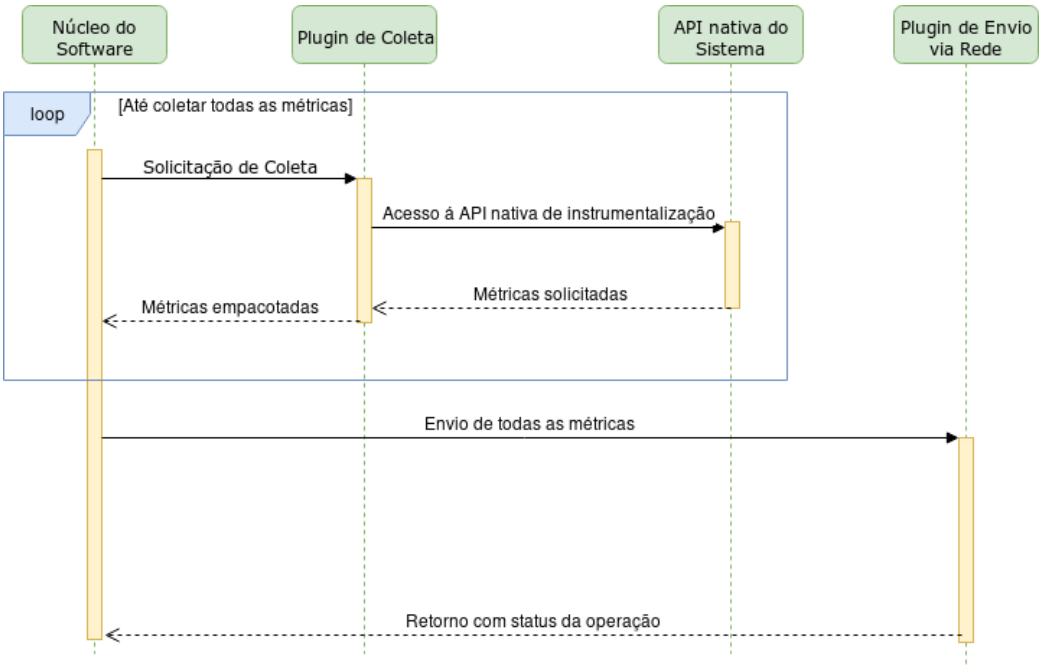
Fonte: Autoria própria

Nas máquinas a serem monitoradas, é implantado o *software* de coleta. O mesmo tem um único papel: o de coletar as métricas selecionadas (utilização de CPU, utilização de memória RAM, espaço em disco, leituras e escritas em disco e utilização de rede), e em um intervalo definido de tempo, enviar as mesmas em requisições feitas pela rede. Isto implica na obrigatoriedade da presença de uma placa de rede e com acesso à mesma rede em que o gerente está presente.

Este tipo de *software* (coleta de métricas) normalmente é composto por módulos (ou *plugins*) independentes: os módulos de coleta fazem a coleta de cada métrica, e elas são enviadas ao banco de dados pelos módulos de escrita (ou envio).

Buscou-se por soluções de coleta que possuem compatibilidade com protocolos comumente empregados em ferramentas de armazenamento de dados, de forma a evitar quaisquer *overheads* e necessidade de configurações extras causados por transformação de dados. A Figura 13 demonstra o processo de coleta de métricas, considerando a estrutura básica de um *software* com este propósito.

Figura 13 - Diagrama de sequência da coleta de métricas



Fonte: Autoria própria

Em um intervalo regular, o *software* de coleta faz requisições à API nativa do sistema operacional, responsável pela instrumentalização do mesmo, para a obtenção das métricas. Ao término deste processo, as mesmas são utilizadas para compor um arquivo ou conjunto de arquivos, os quais trafegam pela rede até a máquina de destino, onde encontra-se o *software* de armazenamento. A requisição retorna então um código de status indicando o sucesso ou falha da operação.

O *software* de armazenamento comporta dados de todos os agentes em utilização, mantendo-os separados. Durante a etapa de implementação, foram feitos *benchmarks* para atestar a capacidade da ferramenta em receber dados de múltiplos agentes, além de ter sido mensurada a eficiência da mesma com diferentes intervalos de coleta a fim de selecionar um intervalo eficiente.

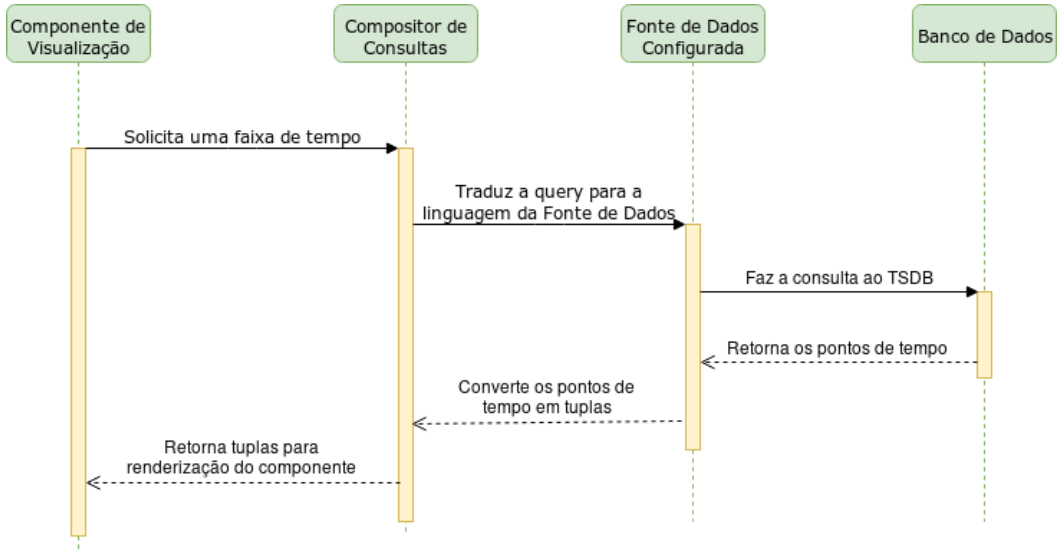
Ao *software* de armazenamento, é integrada uma ferramenta de visualização analítica. A mesma faz a leitura das métricas no banco de dados, e exibe-as em um formato de fácil

compreensão. Visto que ambas ferramentas estão na mesma máquina, o tempo de atraso na execução de consultas pode ser desprezado, permitindo a manipulação de métricas de forma mais responsiva.

Há de se considerar, porém, que ambas ferramentas estão fazendo uso dos mesmos recursos computacionais. Desta forma, na seleção de *softwares* foi dada preferência por ferramentas com um baixo consumo de recursos em casos onde houve uma equivalência entre concorrentes.

A Figura 14 demonstra, através de um diagrama de sequência, a interação e o fluxo de dados entre as ferramentas de armazenamento e de visualização.

Figura 14 - Diagrama de sequência da ferramenta de visualização



Fonte: Autoria própria.

A ferramenta de visualização possui um componente de visualização para cada métrica a ser exibida. Comumente, este componente é um gráfico da métrica em função do tempo, porém é comum a utilização de outras estruturas, como *heatmaps*, indicadores (semelhantes a manômetros) e listas.

Cada componente possui uma *query* pré-definida, a qual é complementada com a faixa de tempo e executada no banco de dados. No caso da existência de resultados, eles são retornados para a ferramenta de visualização em seu formato nativo (pontos de tempo) e são convertidos em tuplas, que podem facilmente ser utilizadas para então renderizar os gráficos dos componentes.

O processo de consulta dos dados à ferramenta de armazenamento repete-se toda vez em que o intervalo de tempo configurado na ferramenta de visualização for alterado, podendo ou não ter dados novos a serem utilizados, dependendo do intervalo da coleta de métricas.

3.2 CRITÉRIOS PARA SELEÇÃO DE FERRAMENTAS

Os critérios a serem considerados para a escolha de cada tipo de ferramenta foram escolhidos levando em consideração o escopo, funcionalidades desejadas e o contexto de implementação e aplicação. As ferramentas a serem selecionadas devem atender a todos os critérios de seu respectivo tipo.

3.2.1 Critérios para seleção de ferramentas de coleta

As ferramentas de coleta, ou *softwares* agentes, devem obrigatoriamente contemplar todos os requisitos abaixo:

a) Compatibilidade com distribuições Linux: É necessário que o *software* de coleta possua uma versão compatível com o sistema operacional;

b) Licença *open-source*: A licença deve possuir licença GPL, MIT ou Apache 2.0;

c) Gratuidade: O *software* deve ser gratuito;

d) Suporte a métricas: O *software* deve possibilitar a coleta de todas as métricas selecionadas, seja nativamente ou por meio de *plugin* oficial. As métricas são: uso de CPU, uso de memória RAM, espaço de armazenamento, utilização de disco (taxas de escrita e leitura), e uso de rede;

e) Envio de métricas para ambiente externo em um intervalo de tempo definido: O *software* deve não apenas coletar, como também possibilitar o envio de métricas para o ambiente externo (outra máquina), o que se dá por meio de rede, seja nativamente ou por meio de *plugin* oficial. O intervalo de tempo entre os envios deve poder ser configurado.

f) A granularidade dos dados deve ser configurável: O intervalo de tempo entre cada coleta deve poder ser escolhido;

g) Compatibilidade com o destino dos dados: Deve ser observada a compatibilidade da ferramenta de coleta com a ferramenta de armazenamento, de forma

a evitar a necessidade de implementações para transporte e transformação de dados, o que pode impactar negativamente no desempenho da integração;

h) Baixo número de dependências: O número de dependências de *software* deve ser o menor possível, de forma a facilitar a instalação e configuração, e não ter quaisquer impactos negativos na máquina a ser monitorada.

3.2.2 Critérios para seleção de ferramentas de armazenamento

De acordo com as necessidades da proposta, a ferramenta de armazenamento foi executada em um ambiente externo ao(s) elemento(s) monitorado(s). As ferramentas concorrentes devem atender obrigatoriamente aos seguintes critérios:

a) Compatibilidade com distribuições Linux: É necessário que o *software* possua uma versão compatível com o sistema operacional;

b) Licença *open-source*: A licença deve possuir licença GPL, MIT ou Apache 2.0;

c) Gratuidade: O *software* deve ser gratuito;

d) Múltiplas origens: Permitir o armazenamento de métricas de mais de um elemento de rede em monitoramento simultaneamente;

e) Retenção customizada: Permitir a configuração de políticas de retenção de dados, de forma a poder limitar o tempo em que dados ficam armazenados;

f) Facilidade: Deve possuir uma linguagem de consultas intuitiva;

g) Conectividade com ambiente externo: Permitir a conexão a partir de ambientes externos à máquina hospedeira.

3.2.3 Critérios para seleção de ferramentas de visualização

As ferramentas de visualização devem obrigatoriamente atender a todos os requisitos abaixo:

a) Compatibilidade com distribuições Linux: É necessário que o *software* possua uma versão compatível com o sistema operacional;

b) Licença *open-source*: A licença deve possuir licença GPL, MIT ou Apache 2.0;

c) Gratuidade: O *software* deve ser gratuito;

d) Interface WEB: Possibilitar a utilização através de navegadores, permitindo uso em qualquer plataforma;

e) Customizável: Permitir a criação, alteração e exclusão de elementos de visualização (gráficos e tabelas);

f) Alertas: Possibilitar a configuração de alertas referentes a métricas específicas, os quais enviam avisos para um canal de comunicação previamente definido;

g) Compatibilidade com a origem dos dados: Deve ser observada a compatibilidade com a ferramenta que fornece os dados, evitando a necessidade de desenvolvimento de scripts para o transporte e transformação de dados, o que pode impactar negativamente no desempenho da integração.

3.3 FERRAMENTAS DE COLETA DE MÉTRICAS

A pesquisa por ferramentas de coletas de métricas foi feita em sites como Stackshare[1] e TechGenix[2], que analisam e comparam ferramentas de gerência de infraestrutura, além de prover conteúdo informativo para profissionais de TI no que tange instalação, configuração, manutenção e otimização de redes. Também foram realizadas pesquisas em artigos como Ward e Barker (2014) e Boncea, Zamfiroiu e Bacivarov (2018). Foram analisadas três ferramentas: Collectd, Telegraf e NetData.

Ward e Barker (2014, p. 10) afirmam que tanto em *cloud* como em ambientes convencionais, o Collectd em conjunto com ferramentas adicionais pode compôr um conjunto de tecnologias simples, porém efetivo. De acordo com Boncea, Zamfiroiu e Bacivarov (2018, p. 17), o Telegraf, componente do *stack* TICK da empresa InfluxData, possui alta capacidade de integração. Ainda, de acordo com Fedora Magazine (2016), o Netdata é uma ferramenta de monitoramento que abstrai as possíveis dificuldades de configuração e manutenção.

Desta forma, as três ferramentas foram analisadas, de acordo com os critérios previamente definidos.

[1] https://stackshare.io/monitoring-tools
[2] http://techgenix.com/open-source-application-monitoring-tools/

3.3.1 Collectd

O Collectd é uma ferramenta *daemon* de uso gratuito sob licença *open-source* MIT que coleta, transfere e, se configurado para isso, armazena dados coletados de elementos computacionais como servidores e equipamentos de rede.

Foi criado por Florian Forster em 2015, e sua última versão estável foi lançada em outubro de 2018. É compatível com qualquer sistema operacional baseado em UNIX, e devido à sua natureza modular requer muito poucos recursos para executar (COLLECTD, 2019).

O *daemon* em si implementa apenas a infraestrutura para filtragem e tratamento de dados, além de algumas funções auxiliares. A coleta, armazenamento e envio de dados são desempenhadas por *plugins* nativos, os quais podem ser ativados e desativados de acordo com a necessidade do utilizador (GITHUB, 2018) e obedecem um intervalo configurado. Pode-se classificar os *plugins* nativos de coleta em três categorias:

- *Plugins* de sistema operacional: coletam métricas a nível de sistema operacional, como CPU, memória, disco e rede, dentre outros. Não são multiplataforma, sendo necessário haver um *plugin* de cada métrica para cada sistema operacional;
- *Plugins* de aplicações: coletam métricas referentes a *softwares* como servidores Apache, Nginx, MySQL ou MariaDB. Exigem algum tipo de instrumentalização dos *softwares*, como API ou *logs*, mas são multiplataforma;
- *Plugins* para tarefas específicas: os que não se encaixam nas categorias anteriores, como *plugin* para execução de *scripts* ou *plugin* para rastreamento de pacotes SNMP na rede.

Já os *plugins* de escrita possuem duas classificações distintas:

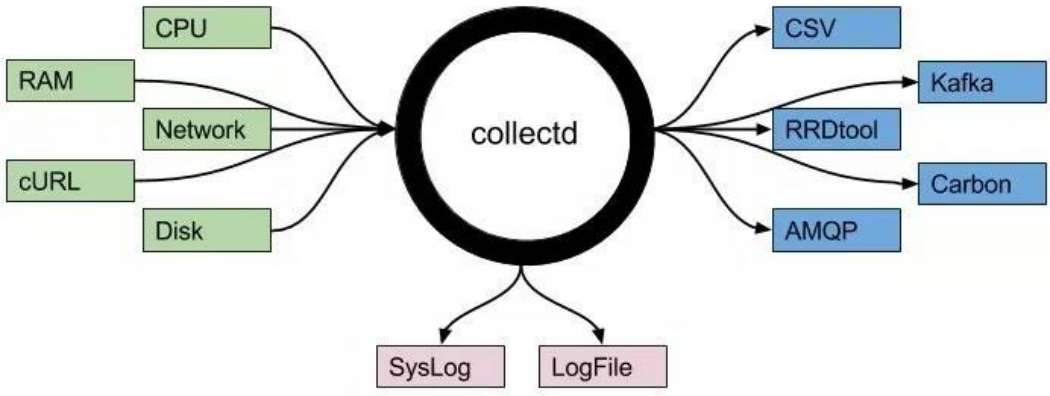
- *Plugins* de armazenamento: armazenam dados coletados na própria máquina, geralmente em RRD;
- *Plugins* de envio: armazenam dados coletados em ambiente externo através do envio via rede, como em banco de dados PostgreSQL, MongoDB, Redis, dentre outros.

Todas as métricas selecionadas possuem *plugins* nativos para a coleta de dados (uso de CPU, uso de memória RAM, espaço de armazenamento, utilização de disco, e uso de rede).

Referente a compatibilidade com *plugins* de envio, é possível enviar dados para ambientes externos como InfluxDB, Graphite e Prometheus.

A Figura 15 demonstra o aspecto modular do Collectd, com *plugins* de coleta e *plugins* de armazenamento (ou escrita).

Figura 15 - Esquema de utilização de plugins com Collectd



Fonte: https://codeblog.dotsandbrackets.com/host-monitoring-with-collectd/ Acesso em: 16 abril 2019

A instalação é simples, e normalmente é feita através dos gerenciadores de pacotes dos sistemas operacionais. Se isto não for viável, é possível baixar o código-fonte e compilar. O mesmo se aplica aos *plugins* (GITHUB, 2018). Não há nenhuma dependência, porém alguns *plugins* específicos podem necessitar a instalação de alguma biblioteca. A lista de dependências é citada na documentação, no site do desenvolvedor do *plugin* (COLLECTD, 2019).

3.3.2 Telegraf

O Telegraf é uma das ferramentas desenvolvidas pela InfluxData, empresa especializada em plataformas de monitoramento de dados em séries temporais. Faz parte de um conjunto de ferramentas, mas pode ser utilizado separadamente e de forma gratuita (INFLUXDATA, 2019).

Desenvolvido em Go e sob a licença *open-source* MIT, consiste em um *software* agente que coleta, processa, agrega e escreve métricas. A arquitetura, semelhante ao Collectd, baseia-se em *plugins* de fácil instalação e configuração, o que proporciona um baixo *footprint* e fácil customização (GITHUB, 2019).

A instalação pode não ser tão simples: utiliza-se do gerenciador de pacotes para instalar o pacote Golang, e baixa-se e compila-se o agente com o gerenciador de pacotes da

linguagem. Caso isso não seja possível, os pacotes .dev e .rpm estão disponíveis para *download* (INFLUXDATA, 2019), porém não evitam a necessidade da presença do pacote Golang. Logo, é uma dependência a ser considerada.

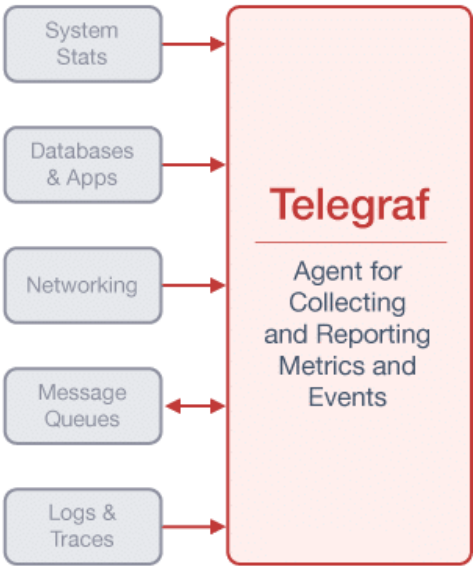
Os *plugins* acompanham o *daemon*, e se classificam em quatro categorias (GITHUB, 2019). São elas:

- Input: *Plugins* que fazem a coleta de métricas do sistema, serviços ou API de terceiros;
- Processor: *Plugins* que convertem e filtram métricas;
- Aggregator: *Plugins* que agregam métricas (calculam médias, valores mínimos, máximos...);
- Output: *Plugins* que armazenam ou enviam métricas para ambientes externos.

Todas as métricas selecionadas (uso de CPU, uso de memória RAM, espaço de armazenamento, utilização de disco e uso de rede) possuem *plugins* nativos para a coleta de dados, e é possível enviar tais dados para ambientes externos (como Graphite, InfluxDB ou Prometheus) por meio de *plugins* específicos. O intervalo de coleta também é configurável através do arquivo de configuração principal do *software*.

A Figura 16 exibe um esquema simplificado das possibilidades de coleta de métricas de sistema.

Figura 16 - Esquema simplificado de entrada de dados no Telegraf



Fonte: https://www.influxdata.com/time-series-platform/telegraf/ Acesso em: 16 abril 2019.

O fato de fazer parte de uma plataforma de ferramentas da mesma empresa desenvolvedora permite a integração com outros *softwares* da mesma plataforma de forma facilitada, através de um protocolo próprio (GITHUB, 2019).

3.3.3 Netdata

Netdata é um *software* de monitoramento sob licença *open-source* GPL v3+ que está disponível para máquinas Linux, FreeBSD e MacOS de forma gratuita (NETDATA, 2019).

Não se trata de um *software* agente como Collectd ou Telegraf: sua instalação *default* possui outros módulos como banco de dados de séries temporais, um visualizador de métricas (Figura 17) e sistema de notificação (NETDATA, 2019). Isto, somado ao fato de que a granularidade padrão é de apenas 1 segundo (a qual pode ser alterada no arquivo de configuração), implicando negativamente no *footprint* do *software*.

O número de dependências é considerável. Pode-se citar autoconf, automake, curl, gcc, git, libmnl-devel, libuuid-devel, lm_sensors, make, MySQL-python, nc, pkgconfig, python, python-psycopg2, PyYAML, zlib-devel, dentre outras. Isto implica na necessidade de considerar aspectos de compatibilidade de cada dependência, o que complica muito a utilização.

Figura 17 - Painel de visualização de métricas do Netdata

Fonte: https://blog.remontti.com.br/2403 acesso em: 16 abril 2019.

O Netdata já vem, por padrão, com todos os módulos de coleta ativados, o que inclui as métricas selecionadas (uso de CPU, uso de memória RAM, espaço de armazenamento,

utilização de disco, e uso de rede). Contempla não somente as métricas de sistema como também de *softwares* de terceiros, o que acontece automaticamente (GITHUB, 2019).

O motivo pelo qual o Netdata pode ser utilizado como agente é o seu modo *slave*. Neste modo os recursos de banco de dados, visualização e alertas são desativados, diminuindo drasticamente os recursos utilizados e permitindo o *streaming* de dados para um ambiente externo, semelhante aos outros *softwares* da mesma categoria. Isto pode ser feito facilmente através de arquivos de configuração.

Dentre os *backends* de armazenamento compatíveis, constam Graphite, Prometheus, InfluxDB, bem como bancos de dados relacionais, orientados a documentos, e outros TSDBs.

O ambiente externo a receber dados pode ser uma instância Netdata no modo *master*, instalada em outra máquina, ou ainda bancos de dados de séries temporais, bancos de dados de documentos, qualquer *software* com interfaces *plaintext* ou telnet como OpenTSDB, InfluxDB, KairosDB, dentre outros (NETDATA, 2019).

3.4 FERRAMENTAS DE ARMAZENAMENTO DE MÉTRICAS

As ferramentas de armazenamento de métricas são, em sua maioria, bancos de dados de séries temporais, ou TSDB (*Time-Series Database*). Isto significa que eles armazenam dados de alguma ocorrência usando o tempo como base e tornando muito mais fácil e menos custoso fazer buscas temporais (LIU; YUAN, 2019). É comum que a funcionalidade de alertas esteja presente em alguns TSDBs. Para esta análise, foram consideradas apenas as capacidades de reter e servir dados. Foram analisadas três ferramentas: InfluxDB, Graphite e Prometheus.

O InfluxDB é, sem dúvida, o banco de dados de série temporais mais popular no mercado atualmente (DB-ENGINES, 2019). Foi desenvolvido para suportar altas taxas de escrita e carga de acessos, com foco no armazenamento de métricas para IoT (*Internet of Things*), sensores e métricas de aplicações (BONCEA; ZAMFIROIU; BACIVAROV, 2018, p. 17).

RIAZ et al (2017, p. 749) afirma que dentre os TSDBs existentes, o Graphite está ganhando popularidade e deve ser examinado se, dentre os requisitos, constar a necessidade de compatibilidade com sistemas operacionais Linux.

Zaitsev (2018) analisa a segunda versão do Prometheus e destaca a grande melhora em performance de escrita, performance de *queries* e eficiência no uso de recursos, características que são fundamentais para este projeto de integração.

Taherizadeh et al (2018, p. 24) analisa e compara estes *softwares* e ferramentas que fazem uso dos mesmos, citando aspectos funcionais da utilização e características que agregam valor em diferentes tipos de ambientes.

3.4.1 InfluxDB

Da mesma forma que o Telegraf, o InfluxDB faz parte da plataforma desenvolvida pela InfluxData. Trata-se de um banco de dados de séries temporais, desenvolvido em Go e sob licença *open-source* MIT. De acordo com o ranking DB-Engines (2019), é o banco de dados de séries temporais mais utilizado, por uma ampla margem.

É compatível com distribuições Linux, além de OS X, e a instalação é simples, sendo os arquivos de *download* disponibilizados no site do desenvolvedor de forma gratuita (INFLUXDATA, 2019).

As estruturas de dados são compostas por medidas, séries e pontos. Cada ponto consiste em vários pares de chave e valor, que quando agrupados, formam uma série, e séries agrupadas por um identificador formam uma medida (GITHUB, 2019). No que tange tipagem, os valores podem ser inteiros de 64 bits, pontos flutuantes de 64 bits, *strings* ou booleanos. Independente do tipo, os pontos são sempre indexados por seu *timestamp*.

Através do 'Line', um protocolo *plaintext*, o InfluxDB aceita conexões de máquinas remotas via protocolo HTTP para efetuar operações (INFLUXDATA, 2019). Tais operações (leitura e gravação) são executadas por meio de uma linguagem denominada InfluxQL, cujas palavras reservadas estão listadas na Figura 18 e permitem ter uma noção básica da sintaxe.

Figura 18 - Lista de palavras reservadas da linguagem InfluxQL

ALL	ALTER	ANY	AS	ASC	BEGIN
BY	CREATE	CONTINUOUS	DATABASE	DATABASES	DEFAULT
DELETE	DESC	DESTINATIONS	DIAGNOSTICS	DISTINCT	DROP
DURATION	END	EVERY	EXPLAIN	FIELD	FOR
FROM	GRANT	GRANTS	GROUP	GROUPS	IN
INF	INSERT	INTO	KEY	KEYS	KILL
LIMIT	SHOW	MEASUREMENT	MEASUREMENTS	NAME	OFFSET
ON	ORDER	PASSWORD	POLICY	POLICIES	PRIVILEGES
QUERIES	QUERY	READ	REPLICATION	RESAMPLE	RETENTION
REVOKE	SELECT	SERIES	SET	SHARD	SHARDS
SLIMIT	SOFFSET	STATS	SUBSCRIPTION	SUBSCRIPTIONS	TAG
TO	USER	USERS	VALUES	WHERE	WITH
WRITE					

Fonte: https://docs.influxdata.com/influxdb/v1.7/query_language/spec/ Acesso em: 16 abril 2019.

Este TSDB permite a criação de *queries* contínuas, as quais executam periodicamente em dados em tempo real e armazenam os resultados em uma medida específica. Ainda, é possível definir políticas de retenção de dados (definindo uma "idade" máxima para os dados armazenados) ou até mesmo reduzir sua granularidade (através de operações de agrupamento) a fim de diminuir o número de registros armazenados (SHAEDEL, 2018).

3.4.2 Graphite

Graphite é um *software open-source* sob licença Apache 2.0, cuja finalidade é o monitoramento de elementos computacionais, independente do contexto. Esta ferramenta desempenha duas funções básicas: armazena dados temporais numéricos, e renderiza gráficos em demanda (GRAPHITE, 2019).

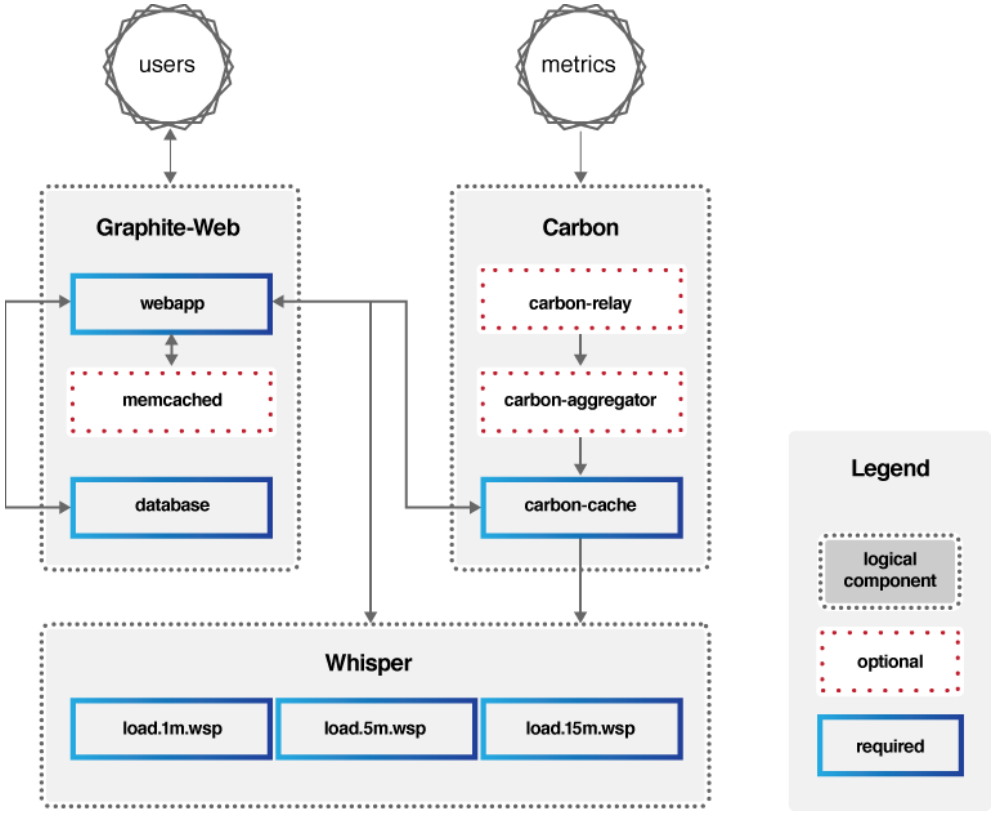
De acordo com o desenvolvedor, qualquer sistema baseado em UNIX pode executar o *software*, porém há de se considerar que é necessária a compatibilidade de todas as dependências. Dentre elas Python 2.7 (tendo suporte experimental a Python 3), Django, algumas bibliotecas python como cairocffi, django-tagging, pytz, scandir, além de ser necessário um servidor WSGI, como Apache, gunicorn ou uWSGI. Para o uso de todas as funcionalidades, ainda se faz necessária a instalação de outros módulos python, como cache python-memcache, python-ldap, txamqp, python-rrdtool, dentre outros (GRAPHITE DOCS, 2018).

Sua arquitetura (Figura 19) consiste em três componentes distintos (DAVIS, 2012):

- Carbon: um serviço de alta performance que recebe dados em séries temporais;
- Whisper: Uma biblioteca de banco de dados para armazenar séries temporais;
- Graphite-web: Uma interface WEB munida de API para renderização de gráficos.

O Graphite aceita conexões tanto da máquina local como de máquinas remotas, através dos protocolos TCP ou UDP. Utiliza-se um formato próprio de dados em *plaintext*, e as métricas coletadas são recebidas pelo Carbon, que prepara e armazena os dados em um ou vários Whispers. O usuário pode interagir com os dados armazenados via Graphite-web, ou pode consumir os dados via API, a qual suporta CSV, XML e JSON (GRAPHITE DOCS, 2018).

Figura 19 - Arquitetura do software Graphite



Fonte: https://graphiteapp.org/#integrations Acesso em: 18 abril 2019

Cada métrica é armazenada no Whisper como um ponto flutuante de dupla precisão (*double*), sendo que cada valor é pareado com um *timestamp*. Caso um valor incompatível seja submetido, o mesmo é convertido (GRAPHITE DOCS, 2018).

A retenção é definida pelo número de pontos de dados, ou por idade máxima, sendo possível agregar dados (através de funções como média, soma, máximo, mínimo ou último) para diminuir a granularidade e espaço ocupado (DAVIS, 2012).

3.4.3 Prometheus

Prometheus é um *software* de monitoramento e alertas *open-source* sob licença Apache 2.0, compatível com sistemas Linux, BSD e Windows (PROMETHEUS, 2019). Suas principais funcionalidades são o armazenamento de métricas em séries temporais, a manipulação de tais métricas através de uma linguagem própria denominada PromQL, o disparo de alertas e a grande capacidade de integrações (ELLIS, 2018).

O núcleo do *software* permite a configuração de federações hierárquicas, eliminando os riscos de pontos únicos de falha além de descentralizar dados, e de federações *cross-service*,

onde um serviço busca dados de outro serviço em outro servidor, possibilitando consultas e alertas em ambos (CARTER, 2018).

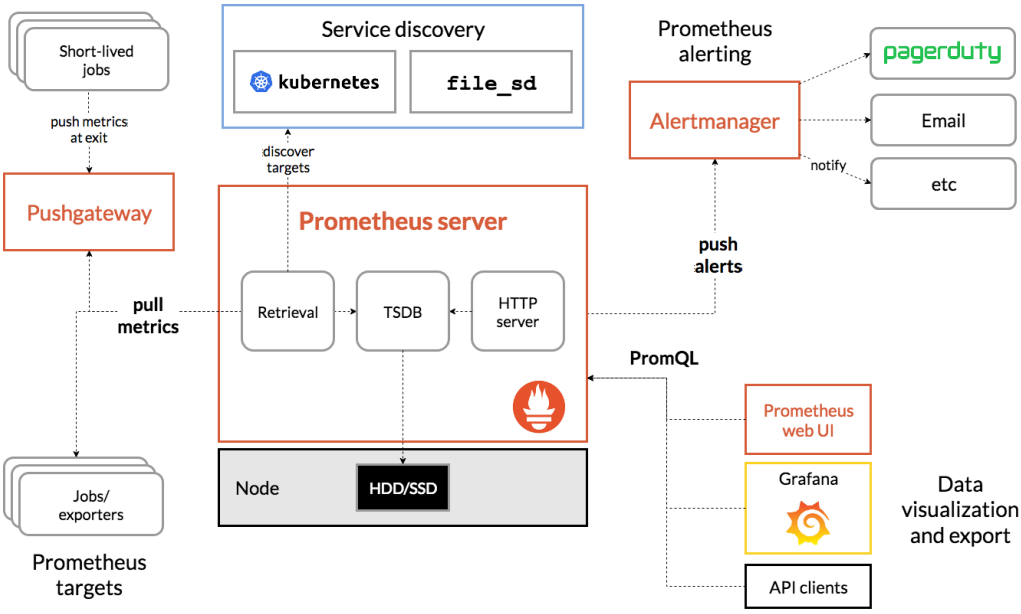
A instalação pode ser feita compilando o código-fonte, ou através da instalação de containers Docker referentes ao componente a ser instalado.

Sua arquitetura (Figura 20) é composta de componentes independentes (KHAN, 2019), que podem ser instalados de forma separada:

- *Core*: O núcleo do *software*, responsável pelo armazenamento de séries temporais e manipulação das mesmas através da linguagem PromQL;
- AlertManager: Gerenciador de alertas, os quais são recebidos do Core ou de aplicações instrumentadas através de bibliotecas específicas, disponíveis no site do desenvolvedor;
- *Exporters*: Ferramentas para a conversão, em *batch* ou tempo real, de dados de outros *softwares* de armazenamento em séries temporais, como Consul, Graphite, Blackbox, dentre outros;
- Pushgateway: Ferramenta que permite tarefas efêmeras e *batches* expor suas métricas ao Core. Visto que este tipo de tarefa não persiste por tempo o suficiente para ser monitorado, ele pode simplesmente registrar as métricas em um PushGateway, o qual se encarrega de integrar com o Core;
- *Libraries*: Conjunto de bibliotecas em diversas linguagens de programação para instrumentação de aplicações. Há bibliotecas oficiais para Go, Java, Python e Ruby, e várias não oficiais, como NodeJS, PHP, Elixir, Rust, dentre outras.

O Prometheus permite conexões tanto da máquina local como de máquinas remotas. Isto pode se dar através de exporters, que agem como servidores para aceitar requisições, ou através de uma API HTTP, na qual envia-se como parâmetros a *query* a ser executada, *timestamps*, e *timeout* (TAHERIZADEH, 2018).

Figura 20 - Arquitetura e componentes do Prometheus



Fonte: https://prometheus.io/docs/introduction/overview/ acesso em: 18 abril 2019.

Referente a retenção de dados, o padrão é a remoção de dados mais antigos que 15 dias através da substituição por dados mais recentes, porém isto é configurável através de parâmetros de execução.

Ainda há uma interface WEB, porém seu único propósito é servir como um *browser* de expressões para a programação de *queries* PromQL.

3.5 FERRAMENTAS DE VISUALIZAÇÃO DE MÉTRICAS

As ferramentas de visualização de métricas permitem a transformação de dados em séries temporais em informações, convertendo tais dados para um formato compreensível e fácil de analisar.

Foram analisadas três ferramentas: Grafana, Kibana e Chronograf.

O Grafana é uma ferramenta rica em formas de visualização, estilos, interatividade e possui excelente documentação *online* (NURGALIEV; KARAVAKIS; AIMAR, 2016, p. 8).

Já o Kibana, o qual embora tenha sido criado para ser utilizado de forma conjunta com o ElasticSearch e tenha foco na visualização de dados de *logs*, também permite a visualização de dados em séries temporais (NURGALIEV; KARAVAKIS; AIMAR, 2016, p. 10).

Por último, foi analisado o Chronograf, que quando utilizado em conjunto com InfluxDB (do mesmo desenvolvedor) permite a utilização de recursos que aumentam o nível de interoperabilidade da integração, além de ser fácil de usar e manter tanto os componentes de

visualização como os alertas e regras de automação (BONCEA; ZAMFIROIU; BACIVAROV, 2018, p. 5).

Boncea, Zamfiroiu e Bacivarov (2018, p. 16-21) também analisam as três ferramentas, citando principalmente os aspectos referentes a integração de dados para com fontes de dados externas, como Prometheus, Graphite e outros TSDBs.

3.5.1 Grafana

O Grafana é a ferramenta de visualização de dados analíticos mais utilizada atualmente. Inicialmente criado por Torkel Ödegaard, atualmente é mantido pela Grafana Labs sob licença *open-source* Apache 2.0, e seu uso é gratuito quando utilizado em infraestrutura própria. Ao utilizar a versão *cloud*, distribuída no modelo SaaS, faz-se necessário o pagamento (GRAFANA, 2019).

É compatível com diversos sistemas Linux, além de possuir versões para Windows, Mac OS X, e pode ser instalado através do *download* direto do pacote, através de um repositório próprio, ou através de um container docker. Após instalado, é acessível através de qualquer navegador, visto que sua interface é um painel *online* (Figura 21) (TAHERIZADEH, 2018).

No que tange fontes de dados, o Grafana pode efetuar consultas nativamente em bancos de dados Graphite, Prometheus, InfluxDB, ElasticSearch, Stackdriver, OpenTSDB, e uma série de bancos relacionais (é possível a instalação de *plugins* oficiais para permitir outras fontes de dados). Através dos dados consultados, pode-se criar gráficos customizados, tabelas, indicadores e até mesmo *heatmaps* (GRAFANA, 2019).

Figura 21 - Exemplo de painel do Grafana com múltiplos gráficos

Fonte: https://www.weave.works/docs/tutorials/core/weave-cloud-monitor-and-grafana/ acesso em: 19 abril 2019

Outra grande funcionalidade é o componente de alertas. Regras previamente definidas são avaliadas no *backend* da aplicação, através de comparações com os dados recebidos periodicamente. O mais comum é a definição de uma expressão juntamente com um intervalo de tempo: se a expressão for verdadeira durante todo o tempo definido, o alerta é acionado. Isto reduz a possibilidade do recebimento de falsos-positivos.

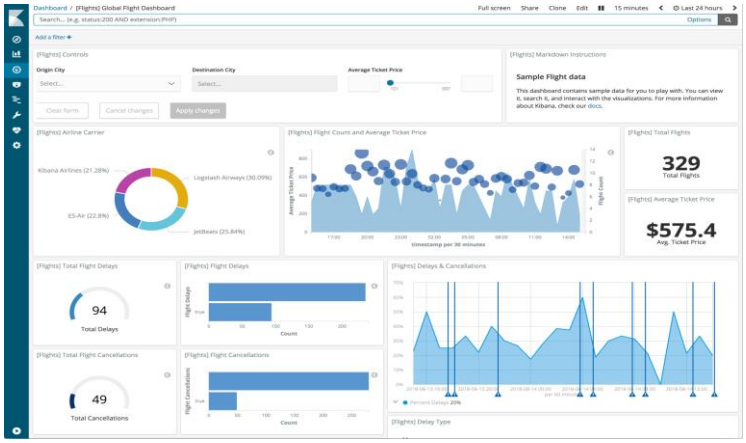
Dentre os canais de comunicação para o recebimento de alertas, há a compatibilidade com envio de *e-mails*, chamadas de *webhooks*, Slack, Telegram, Kafka, Pagerduty, entre outros (GRAFANA, 2019).

3.5.2 Kibana

O Kibana é uma ferramenta de visualização desenvolvida pela Elastic, e faz parte da plataforma ELK (Elastic, Logstash e Kibana). Sob licença *open-source* Apache, seu uso é gratuito se utilizado em infraestrutura própria e possui para *download* versões para execução em Windows, Mac OS X, Linux. Também é possível executar em um *container* docker (ELASTIC, 2019).

A ferramenta permite a composição de gráficos, *logs*, mapas e outros recursos visuais (denominados "visões") construídos sobre conteúdo indexado em um cluster ElasticSearch, e acessíveis através de uma interface WEB (Figura 22) (NURGALIEV; KARAVAKIS; AIMAR, 2016, p. 6). O componente Logstash recebe dados referentes a métricas e transforma em um formato compatível com o ElasticSearch. Este, por sua vez, integra-se ao Kibana fornecendo dados padronizados (APARECIDO, 2017).

Figura 22 - Exemplo de visão do Kibana



Fonte: https://www.elastic.co/guide/en/kibana/6.7/release-highlights-6.4.0.html acesso em: 19 abril 2019.

Claramente, trata-se de uma ferramenta com grande dependência em componentes externos do mesmo desenvolvedor. A capacidade de disparo de alertas não está disponível no Kibana; porém, está no ElasticSearch (ELASTIC, 2019).

Referente à compatibilidade para com TSDBs, o Logstash possui alguns *plugins* oficiais para a recepção de dados. Porém, o único TSDB disponível dentre os analisados é o Graphite (ELASTIC, 2019).

3.5.3 Chronograf

O Chronograf é um visualizador de dados desenvolvido pela InfluxData. Distribuído sob licença *open-source* GPL e de forma gratuita, o *software* possui versões compatíveis com Mac OS X e distribuições Linux. Não é necessária a instalação, visto que basta efetuar o *download* do *software* e executá-lo. Para acessar, utiliza-se um navegador, visto que é composto por uma interface WEB (INFLUXDATA, 2019).

O *software* permite a criação de paineis (ou *dashboards*, como são denominados) com diversas formas: gráficos de linhas, barras, pizza, tabelas, indicadores e até mesmo tabelas (Figura 23) (SIMMONS, 2018).

Figura 23 - Exemplo de dashboard no Chronograf

Fonte: https://docs.influxdata.com/chronograf/v1.7/guides/create-a-dashboard/ acesso em 23 abril 2019.

Só é possível utilizar bancos de dados InfluxDB como fontes de dados. Porém, a integração ocorre de forma quase que automatizada: o Chronograf detecta o esquema dos bancos de dados e cria paineis automaticamente (FARMER, 2018).

É possível configurar alertas, porém isto é feito através de um componente extra a ser instalado de forma separada: o Kapacitor. O mesmo permite a criação de alertas, tarefas de ETL (*Extract, Transform, Load*), e normalização de dados. Da mesma forma que o Grafana, os canais de comunicação suportados são diversos: Slack, *e-mail*, Telegram, chamadas *webhooks*, Kafka, dentre outros (INFLUXDATA, 2019).

3.6 CONSIDERAÇÕES DO CAPÍTULO

Considerando a finalidade das ferramentas selecionadas, foram analisados três *softwares*, cada um pertencente a um dos três tipos de ferramentas, através de critérios previamente definidos neste trabalho. Ao mesmo tempo, foram consideradas suas capacidades de integração no contexto de ambiente de rede.

Quanto à ferramenta de coleta de métricas, foram analisados três agentes: Collectd, Telegraf e Netdata. O Quadro 2 ilustra o resultado.

Quadro 2 - Análise de critérios para escolha de ferramenta de coleta

Critérios	**Collectd**	**Telegraf**	**Netdata**
Compatibilidade com Linux	Sim	Sim	Sim
Licença *open-source*	Sim	Sim	Sim
Gratuidade	Sim	Sim	Sim
Suporte às métricas	Todas	Todas	Todas
Envio de métricas para ambiente externo	Sim	Sim	Necessário configurar modo *slave*
Configuração de granularidade	Sim	Sim	Sim
Compatibilidade com TSDBs	InfluxDB, Graphite, Prometheus...	InfluxDB, Graphite, Prometheus...	InfluxDB, Graphite, Prometheus...
Dependências	Nenhuma	1 (Golang)	16 (libuuid-devel, lm_sensors, make, MySQL-python, python, python-psycopg2, PyYAML, zlib-devel, dentre outras)

Fonte: Autoria própria.

Todos os *softwares* analisados estão de acordo com os critérios avaliados. Porém, algumas características do Netdata não são satisfatórias para o projeto: o fato de ser uma ferramenta que por padrão faz uso dos recursos computacionais para manter tanto o armazenamento como um ambiente de visualização de métricas faz com que seu *footprint* seja mais alto que os das outras alternativas. Isto, somado ao fato de ser necessária a configuração de um modo slave para o envio de métricas para ambiente externo, não são ideais.

O Telegraf teve uma avaliação satisfatória, porém a necessidade da instalação de dependências de forma manual para o correto funcionamento da ferramenta foi um fato determinante para a exclusão desta opção: agentes devem possuir não apenas um baixo consumo de recursos, como também poucas dependências (ou nenhuma). Isto reduz a possibilidade de problemas de compatibilidade e depreciação de pacotes.

O Collectd foi a ferramenta de coleta selecionada. Além de não possuir dependências, a instalação de *plugins* é transparente e ocorre através de repositórios padrões. O *footprint* é baixo, e a configuração é simples, através de um arquivo *.conf*.

Referente à ferramenta de armazenamento de métricas, a qual será executada juntamente com a ferramenta de visualização analítica, foram analisadas três alternativas: InfluxDB, Graphite e Prometheus. O Quadro 3 demonstra os critérios e resultados da avaliação.

Quadro 3 - Análise de critérios para escolha da ferramenta de armazenamento

Critérios	**InfluxDB**	**Graphite**	**Prometheus**
Compatibilidade com Linux	Sim	Sim	Sim
Licença *open-source*	Sim	Sim	Sim
Gratuidade	Sim	Sim	Sim
Múltiplas origens	Sim	Sim	Sim
Retenção customizada	Sim	Sim, porém é necessário usar o intervalo para definir a regra	Sim (via CLI)
Facilidade de uso da linguagem	Sim (InfluxQL)	Não	Sim (PromQL)
Conectividade com ambiente externo	Sim (Line Protocol/HTTP)	Sim (*Plaintext* via UDP/TCP)	Sim (API HTTP REST)

Fonte: Autoria própria.

O excesso de dependências é o principal ponto negativo do Graphite. Os três módulos que compõem o *software* possuem um procedimento de instalação complexo. Isto, somado à

ausência de uma interface de fácil acesso aos dados, descarta a ferramenta. Faz-se uso de funções que se aplicam em uma métrica a cada vez, ao invés de uma *query language* (QL) própria.

O Prometheus foi considerado uma alternativa satisfatória, porém com uma ressalva: a integração com as ferramentas de coleta analisadas funciona através do *scraping* de dados via interface acessível na máquina monitorada. Isto se dá através de *exporters*, que são basicamente componentes de ETL. A obrigação de disponibilizar uma interface para scraping infere a abertura de portas no *firewall* da máquina monitorada, o que não é desejável.

A ferramenta de armazenamento selecionada foi o InfluxDB. Considerado o TSDB mais maduro disponível no mercado (DB-ENGINES, 2019), ele suporta a recepção de múltiplos agentes, além de permitir a configuração do tempo de retenção de forma global ou customizada para cada banco de dados. Embora haja dependências (Golang), isto é aceitável considerando que a instalação se dará em apenas uma máquina (gerente) e uma única vez.

Por fim, foram analisadas três ferramentas de visualização analítica de dados: Grafana, Kibana e Chronograf. O Quadro 4 demonstra os critérios e resultados da análise.

Quadro 4 - Análise de critérios para escolha de ferramenta de visualização analítica

Critérios	**Grafana**	**Kibana**	**Chronograf**
Compatibilidade com Linux	Sim	Sim	Sim
Licença *open-source*	Sim	Sim	Sim
Gratuidade	Sim	Sim	Sim
Interface WEB	Sim	Sim	Sim
Customizável	Sim	Há limitações	Sim
Alertas	Sim	Não	Não
Compatibilidade com Origens de Dados	Sim (Graphite, Prometheus, InfluxDB, dentre muitos outros)	Não (Aceita apenas ElasticSearch, o qual é alimentado via Logstash)	Apenas InfluxDB

Fonte: Autoria própria.

O Kibana foi aprovado na maioria dos critérios, porém há fatores determinantes para sua desaprovação neste projeto. O principal é o fato de que ele não aceita como origem de dados quaisquer ferramentas que não são desenvolvidas pela Elastic, seu desenvolvedor. Faz-se necessário utilizar o banco de dados Elasticsearch, e o mesmo apenas aceita dados oriundos do

Logstash, outra ferramenta do mesmo *stack*. Isto aumenta consideravelmente a complexidade da integração, além da carga de trabalho necessária para a transformação de dados. Ainda, neste projeto há a necessidade de emissão de alertas de acordo com um conjunto de regras predefinidas, tarefa essa que o Kibana não consegue desempenhar sem *softwares* externos.

O Chronograf teve uma avaliação boa devido à facilidade de uso, porém isto só é possível devido ao fato de que utiliza-se de uma única origem de dados: o InfluxDB, do mesmo desenvolvedor. É impossível utilizar outra fonte de dados, e também não é possível emitir alertas sem ferramenta externa. A integração com o Kapacitor, ferramenta de alertas da InfluxData, é simples, porém uma quarta ferramenta não faz parte do escopo do projeto.

O Grafana teve sem dúvida a melhor avaliação. Todos os critérios foram atendidos satisfatoriamente. A ampla compatibilidade com diversas origens de dados, e a possibilidade de desenvolver os gráficos com diferentes linguagens de consulta são características muito interessantes. Desta forma, a ferramenta escolhida para a visualização analítica de dados foi o Grafana.

4 PROPOSTA DE SOLUÇÃO

Este capítulo é composto por 7 seções, que descrevem de forma detalhada a solução proposta para que o objetivo deste trabalho tenha sido atingido.

Na primeira seção, foram descritos os requisitos mínimos das ferramentas selecionadas. Na segunda seção, descreveu-se os requisitos funcionais e não funcionais da integração, levando em conta as funcionalidades previamente levantadas. Cada um dos requisitos funcionais foi detalhado em seus respectivos casos de uso na terceira seção.

A arquitetura proposta é descrita na quarta seção, e na quinta seção, descreve-se a metodologia para a validação da solução. A única ferramenta que possui aspectos visíveis é o Grafana, cujos protótipos se encontram na sexta seção. Na sétima seção são descritas as considerações finais da proposta.

4.1 REQUISITOS MÍNIMOS DAS FERRAMENTAS SELECIONADAS

Os requisitos mínimos do InfluxDB em sua última versão (1.7.5) estão especificados no site oficial do desenvolvedor. Eles dizem respeito a casos de utilização com aproximadamente 5 mil escritas por segundo. O Quadro 5 demonstra tais requisitos de *hardware*.

Quadro 5 - Requisitos mínimos considerados para o InfluxDB

Processador	2-4 núcleos
Memória RAM	2-4GB
Armazenamento	5GB, 500 IOPS

Fonte: Autoria própria

No contexto onde este trabalho foi posto em prática, é executada uma média de 30 a 40 escritas por segundo para cada agente em utilização (considerando as métricas selecionadas no capítulo 3), o que reduz consideravemente os requisitos mínimos necessários.

Referente a ferramenta de visualização analítica (Grafana), foi utilizada a última versão estável lançada (6.1.6), e os requisitos mínimos variam conforme a quantidade de paineis e *dashboards* configurados. De forma geral, considera-se os requisitos explícitos no Quadro 6.

Quadro 6 - Requisitos mínimos do Grafana

Processador	1 núcleo
Memória RAM	256MB
Armazenamento	1GB

Fonte: Autoria própria

Observando os requerimentos mínimos das aplicações a serem executadas neste ambiente, além de limitações em investimentos, o ambiente empregado foi um servidor próprio da organização, onde a solução proposta foi implementada para validação. O mesmo se encontra *on premise*, e é administrado pela própria organização. O Quadro 7 exibe as especificações de *hardware* e *software* do mesmo.

Quadro 7 - Especificações do ambiente gerente

Processador	Intel Xeon, arquitetura WolfDale™
Memória RAM	16GB
Armazenamento	2x 2TB HDD em RAID 1
Limitação de banda	Ilimitado
Velocidade de Rede	1GB Ethernet
Link de Internet	50MB *Download* / 10MB *Upload*
Sistema operacional	Debian 9.9 x64 (codinome Stretch)

Fonte: Autoria própria

De forma a evitar conflitos com outras soluções em execução no ambiente, foi feito o uso de virtualização. O Quadro 8 demonstra os recursos reservados para a máquina virtual.

Quadro 8 - Especificações da máquina virtual hospedada

Processador	1 core
Memória RAM	1GB
Armazenamento	15GB
Sistema Operacional	Debian 9.9 x64 (codinome Stretch)

Fonte: Autoria própria

A organização disponibilizou um endereço IP (versão 4) para a instância, porém foi configurado um nome de domínio próprio de forma a facilitar os procedimentos de configuração dos *softwares* agentes nas máquinas a serem monitoradas.

4.2 DEFINIÇÃO DE REQUISITOS

A integração das ferramentas teve por objetivos: oferecer uma plataforma que permita ao profissional utilizador uma maneira de identificar o estado atual de elementos

computacionais monitorados, ao mesmo tempo de fazê-lo de forma rápida e simples, e oferecer recursos que permitam a constatação e identificação de incidentes ligados à utilização de recursos computacionais de forma quase que instantânea, de forma a agilizar uma possível mitigação.

Para tanto, os requisitos funcionais do projeto são:

- RF-001: Permitir efetuar *login* na ferramenta de visualização (Grafana);
- RF-002: Permitir manter usuários cadastrados;
- RF-003: Permitir manter máquinas monitoradas pelo agente selecionado;
- RF-004: Permitir alternar entre métricas de máquinas diferentes;
- RF-004: Permitir a busca de métricas por meio de um seletor de faixa de tempo;
- RF-005: Permitir a manipulação de métricas, aumentando ou diminuindo o nível de detalhamento da visualização;
- RF-006: Permitir a configuração de disparo de alertas via Telegram quando métricas atingem um determinado nível.

Os requisitos não funcionais são:

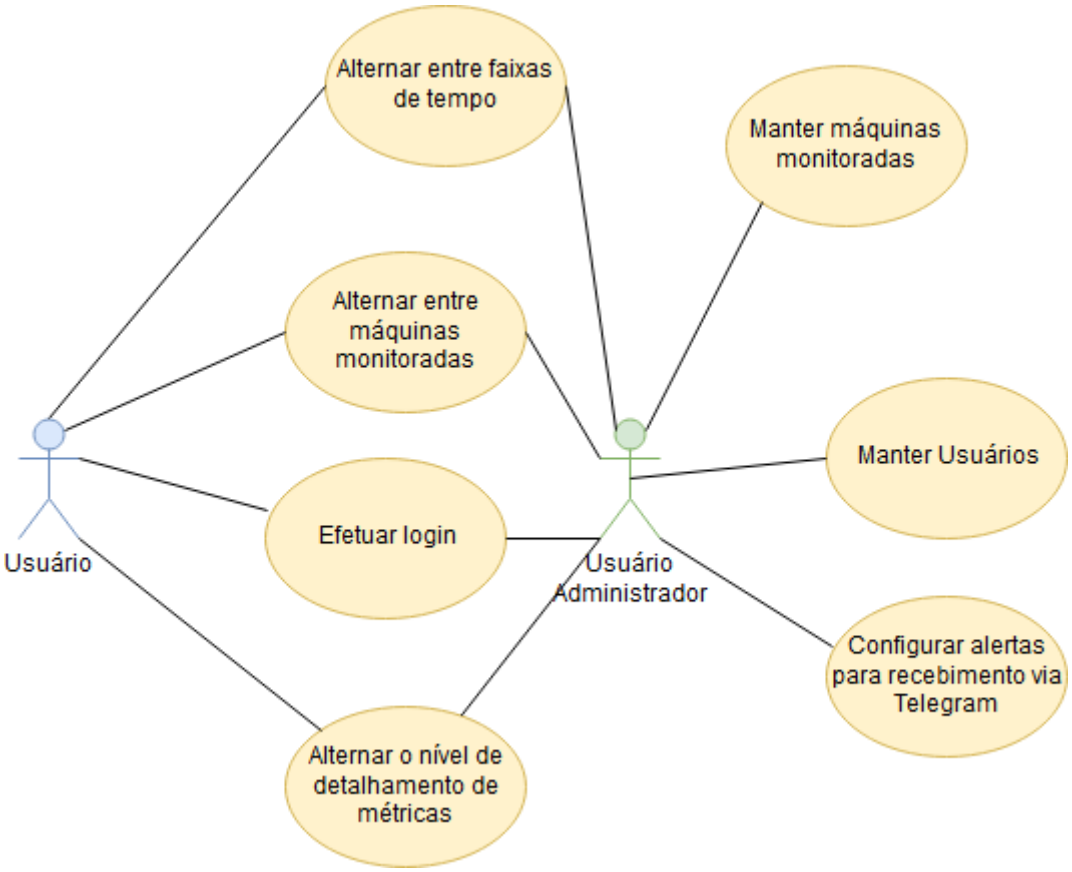
- RFN-001: Permitir o acesso à ferramenta de visualização através de *browsers* de internet, garantindo total compatibilidade de todas as funções com Google Chrome 74[3] e Mozilla Firefox 66[4];
- RFN-002: O agente utilizado nas máquinas monitoradas deve ser compatível com distribuições Linux;
- RFN-003: A GUI (*Graphical User Interface*, ou Interface Gráfica de Usuário) deve ser acessível e de baixa complexidade;
- RFN-004: O tempo de carregamento de cada painel de visualização não deve exceder 4 segundos;
- RFN-005: O acesso à ferramenta deve ser feito apenas por meio de autenticação com usuário e senha cadastrado previamente.

[3] https://www.google.com/intl/pt-BR/chrome/
[4] https://www.mozilla.org/pt-BR/firefox/new/

4.3 DETALHAMENTOS DE CASO DE USO

O diagrama dos casos de uso para a integração dos *softwares* é mostrado na Figura 28. Cada caso de uso deriva de um dos requisitos funcionais previamente levantados, e é detalhado nos subcapítulos seguintes.

Figura 24 - Diagrama de casos de uso



Fonte: Autoria própria

Como é possível ver, há dois atores que interagem com a aplicação: um usuário com permissões de acesso básicas, e um usuário com permissões de administrador. O administrador pode, além de funções de gerenciamento, também exercer funções de usuário básicas.

4.3.1 Efetuar login na ferramenta Grafana

O Quadro 9 demonstra as características do caso de uso de *login* na ferramenta de visualização analítica. Este procedimento exige que o utilizador possua um cadastro efetuado, e conheça seu nome de usuário e senha.

O login exige que seja feito o acesso à ferramenta Grafana, através de um *browser* suportado, utilizando o endereço e porta nos quais foi feita a instalação da aplicação.

Quadro 9 - Caso de uso para “Efetuar login na ferramenta”

Caso de Uso	Efetuar *login* na ferramenta (RF-001)
Descrição	O usuário deve acessar a ferramenta através de um *browser* de internet, e autenticar-se para poder usufruir das funcionalidades
Atores envolvidos	Usuário do Sistema
Pré-condição	O usuário deve possuir um cadastro na ferramenta de visualização analítica
Fluxo Principal	1. O usuário acessa a ferramenta de visualização analítica, digitando o endereço em um *browser* de internet; 2. O usuário digita o *login* e senha nos campos correspondentes e clica em “*Log In*”; 3. A ferramenta autentica a sessão, e o usuário é redirecionado para a tela principal da ferramenta.
Fluxos Alternativos	Item 1: Por algum motivo, a ferramenta se encontra inacessível. 1. Reportar ao administrador do sistema.
	Item 2: O usuário não possui nome de usuário e senha previamente cadastrados. 1. O usuário solicita ao administrador da ferramenta dados para autenticação;
	Item 3: O usuário digita os dados de autenticação de forma incorreta. 1. O sistema exibe um aviso de nome de usuário e/ou senha incorretos.
Pós-condição	Usuário com sessão autenticada à ferramenta de visualização analítica.

Fonte: Autoria própria

4.3.2 Manter usuários

O Quadro 10 demonstra o detalhamento do caso de uso de manter usuários, o qual é responsável pelas operações em usuários que podem acessar a ferramenta de visualização analítica. Este procedimento exige que o utilizador possua um cadastro efetuado, e conheça seu nome de usuário e senha.

Por “manter”, subentende-se as quatro operações básicas: listar, cadastrar, editar e excluir usuários. Um usuário pode pertencer a uma organização, previamente cadastrada, visto que seu nível de acesso pode variar conforme a organização a que pertence.

Quadro 10 - Caso de uso para “Manter usuários”

Caso de Uso	Manter Usuários (RF-002)
Descrição	O usuário administrador cadastra usuários para o *software*, podendo inclusive editar e excluir.
Atores envolvidos	Usuário administrador do Sistema
Pré-condição	O usuário deve possuir um cadastro com nível de administrador na ferramenta de visualização analítica
Fluxo Principal	1. O administrador acessa a listagem de usuários do *software*; 2. O sistema lista todos os usuários cadastrados, independente de nível de acesso; 3. O administrador acessa o cadastro de usuários; 4. O sistema exibe o formulário cadastral; 5. O administrador preenche todos os campos obrigatórios; 6. O administrador submete o formulário; 7. O sistema redireciona o administrador de volta para a lista.
Fluxos Alternativos	Item 1: O administrador não preenche todos os campos obrigatórios. 1. O sistema exibe um aviso de obrigatoriedade e bloqueia.
Pós-condição	Usuário cadastrado, editado ou excluído.

Fonte: Autoria própria

4.3.3 Manter máquinas monitoradas

O Quadro 11 exibe o detalhamento do caso de uso de manter máquinas monitoradas. Neste caso de uso, utilizou-se da funcionalidade de criação de *dashboards* do Grafana para criar um *dashboard* para cada máquina a ser monitorada. Aos mesmos, atribui-se como identificador o *hostname* de sua respectiva máquina. Esta forma de adicionar agentes permite estabelecer parâmetros customizados para cada ambiente, como cores, limites para alertas, e até mesmo paineis, se necessário.

Por “manter”, subentende-se as operações de inserção, edição ou exclusão dos nomes de domínio das máquinas monitoradas.

Um fator importante da administração de *dashboards* que deve ser considerado é que é uma funcionalidade que exige permissões de acesso de administrador na ferramenta Grafana.

Quadro 11 - Caso de uso “Manter máquinas monitoradas”

Caso de Uso	Manter máquinas monitoradas (RF-003)
Descrição	O usuário administrador acessa o gerenciamento de dashboards no painel de monitoramento na ferramenta de visualização analítica, e adiciona um novo dashboard com o título contendo o *hostname* da nova máquina. O mesmo pode ser alterado ou removido se necessário.
Atores envolvidos	Usuário administrador do Sistema
Pré-condição	O usuário deve possuir um cadastro com nível de administrador na ferramenta de visualização analítica; O usuário deve possuir o nível de acesso de edição de *dashboards*; O *dashboard* deve estar com a permissão de edição habilitada.
Fluxo Principal	1. O usuário administrador acessa o *dashboard* principal do monitoramento de servidores; 2. O usuário administrador clica no botão ‘*New Dashboard*’; 3. O usuário administrador cria os paineis desejados; 4. O usuário configura as *queries* e estilos dos paineis criados; 5. O usuário clica em “*Save*” para salvar as alterações.
Fluxos Alternativos	Item 1: O usuário não possui permissões de criação. 1. O usuário solicita a permissão ao administrador; 2. O usuário administrador confere a permissão necessária; 3. O usuário efetua a criação do novo *dashboard*.
Pós-condição	*Dashboard* adicionado, alterado ou excluído.

Fonte: Autoria própria

4.3.4 Alternar entre diferentes máquinas

O projeto contempla o monitoramento de métricas em uma ou mais máquinas, no contexto em que será aplicado. Fez-se necessário que seja possível alternar entre os paineis de exibição de cada máquina monitorada ativa.

Há de se considerar que o *dashboard* possui três modos de visualização: padrão, *Cycle* e *Kiosk*. Os modos *Cycle* e *Kiosk* invisibilizam elementos da interface de visualização fazendo com que os paineis de métricas redimensionem-se, dando mais espaço para os mesmos. Isto teve implicações neste caso de uso, de forma que a alternação entre diferentes máquinas é possível apenas no modo padrão, o qual possui o seletor de *dashboards* (para seleção de máquinas a serem monitoradas). O Quadro 12 exibe o detalhamento referente a este requisito funcional.

Quadro 12 - Caso de uso “Alternar entre máquinas diferentes”

Caso de Uso	Alternar entre máquinas diferentes (RF-004)
Descrição	O usuário poderá alternar para outras máquinas previamente adicionadas à ferramenta, de forma a visualizar suas respectivas métricas.
Atores envolvidos	Usuário do Sistema
Pré-condição	O usuário deve possuir um cadastro com nível de administrador na ferramenta de visualização analítica; O usuário deve ter permissão de, no mínimo, visualização do *dashboard*; O *dashboard* deve possuir, no mínimo, dois *dashboards* cadastrados (cada uma representando uma máquina monitorada).
Fluxo Principal	1. O usuário acessa o *dashboard* “Monitoramento de Servidores”; 2. O usuário clica no grupo de dashboards principal (*General*); 3. Ao clicar na opção desejada do seletor, os componentes do *dashboard* serão renderizados para visualização.
Fluxos Alternativos	Item 1: O usuário não tem permissão de visualizar o *dashboard*. 1. Solicita permissão ao administrador.
	Item 2: O seletor não está visível. 1. O usuário deve verificar se o *dashboard* está em modo *Cycle* ou *Kiosk*; 2. O usuário deve sair destes modos para poder visualizar o seletor.
Pós-condição	Máquina desejada em primeiro plano de visualização.

Fonte: Autoria própria

4.3.5 Alternar entre faixas de tempo

A ferramenta de visualização teve de permitir a seleção de faixas de tempo para as métricas a serem visualizadas. Uma faixa de tempo deve ser composta por um *timestamp* de início e um de fim, sendo o primeiro obrigatoriamente anterior ao segundo.

Semelhante ao caso de uso “Alternar entre diferentes máquinas”, neste caso de uso há implicações no que tange o seletor de faixas de tempo: o mesmo não é visível no modo *Kiosk*, fazendo-se necessário entrar em modo padrão ou *Cycle* para visualizá-lo e utilizá-lo. O Quadro 13 indica o caso de uso referente a este requisito.

Quadro 13 - Caso de uso “Alternar entre faixas de tempo”

Caso de Uso	Alternar entre faixas de tempo (RF-005)
Descrição	O usuário poderá selecionar diferentes faixas de tempo, podendo ser absolutas ou relativas à data/hora atual.
Atores envolvidos	Usuário do Sistema
Pré-condição	O usuário deve possuir um cadastro na ferramenta de visualização analítica; O usuário deve ter permissão de, no mínimo, visualização do *dashboard*.
Fluxo Principal	1. O usuário acessa o *dashboard* de monitoramento de servidores; 2. O usuário clica no campo de seleção de datas e horas; 3. O usuário pode clicar em uma das faixas de data e hora pré-definidas, as quais são relativas à data/hora atual, ou selecionar manualmente um *timestamp* de início e de fim; 4. O usuário clica em “*Apply*”.
Fluxos Alternativos	Item 1: O usuário não possui permissões de acesso ao *dashboard*. 1. O usuário solicita o acesso ao administrador;
	Item 2: O usuário não encontra o seletor no canto superior direito. 1. O usuário verifica se o sistema está em modo *Cycle* ou modo *Kiosk*; 2. O usuário sai do modo *Cycle* ou *Kiosk*, se estiver.
	Item 3: A faixa selecionada é inválida. 1. Os componentes do *dashboard* não exibem dados; 2. O usuário deve selecionar um intervalo de tempo válido.
Pós-condição	Visualização de métricas referentes à faixa de tempo selecionada.

Fonte: Autoria própria

4.3.6 Alternar o nível de detalhamento das métricas

O Quadro 14 descreve o caso de uso de alteração do detalhamento de métricas. Esta funcionalidade permite que, interagindo com os componentes do *dashboard*, seja possível selecionar um período de tempo onde possa ter ocorrido algum incidente. Ao fazer esta seleção em um dos componentes, o tempo selecionado é filtrado em todos os outros componentes do *dashboard*.

Quadro 14 - Caso de uso “Alternar o detalhamento de métricas”

Caso de Uso	Alternar o detalhamento de métricas (RF-006)
Descrição	O usuário pode selecionar um período de métricas a ser visualizado ao interagir com os componentes do *dashboard*, aumentando ou diminuindo o detalhamento.
Atores envolvidos	Usuário do Sistema
Pré-condição	O usuário deve possuir um cadastro na ferramenta de visualização analítica; O usuário deve ter permissão de, no mínimo, visualização do *dashboard*.
Fluxo Principal	1. O usuário acessa o *dashboard* de monitoramento de servidores; 2. O usuário escolhe um dos componentes para efetuar a seleção; 3. O usuário seleciona, através do mouse ou dispositivo apontador em uso, um ponto de início e de fim no componente; 4. O *dashboard* recarrega com as métricas do momento selecionado.
Fluxos Alternativos	Item 1: O usuário não tem permissão de visualização ao *dashboard*. 1. O usuário solicita a permissão de visualização para um administrador.
	Item 4: Os componentes recarregaram sem nenhum indicador de métricas. 1. O período selecionado não possuia nenhum dado disponível; 2. O usuário seleciona um outro período.
Pós-condição	Visualização de métricas referentes ao intervalo de tempo selecionado.

Fonte: Autoria própria

4.3.7 Configurar alertas para Telegram

Este caso de uso diz respeito ao requisito de disparo de notificações de acordo com limites preestabelecidos. Todos os componentes do *dashboard* devem possuir uma função de alerta na qual configura-se um período de avaliação, e caso uma ou várias condições configuradas sejam verdadeiras pelo período configurado, o canal de comunicação configurado (Telegram) dispara mensagens para um contato ou grupo de contatos criado previamente. O Quadro 15 ilustra o caso de uso.

Quadro 15 - Caso de uso “Configurar alertas para Telegram”

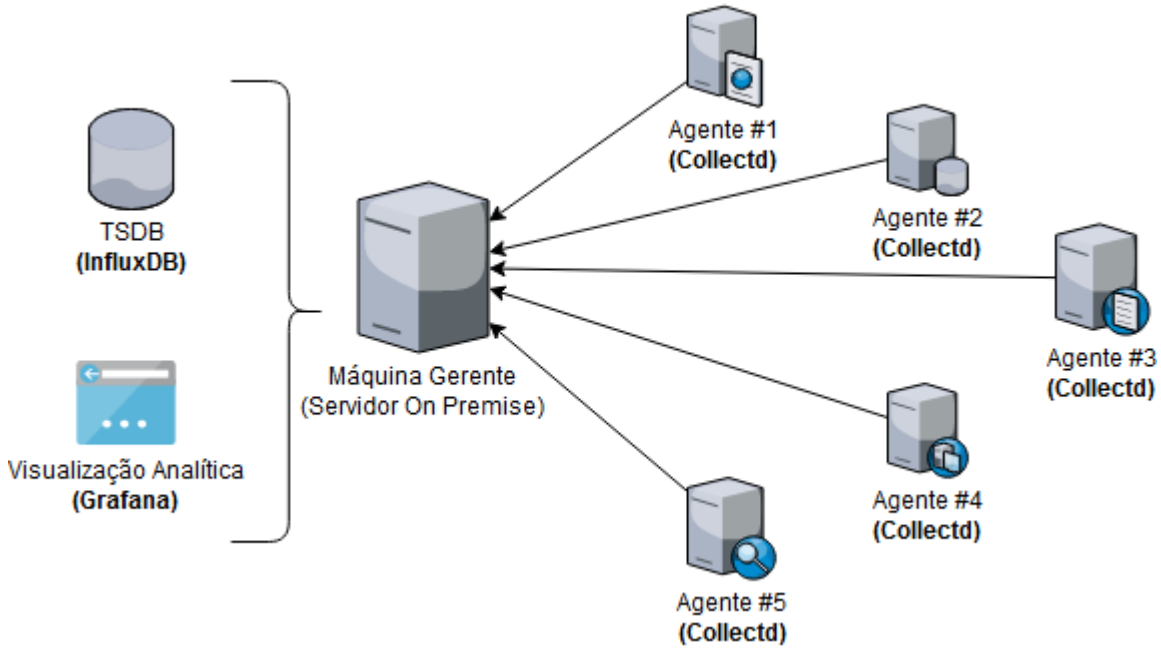
Caso de Uso	Configurar alertas para Telegram (RF-007)
Descrição	O usuário pode selecionar um período de métricas a ser visualizado ao interagir com os componentes do *dashboard*.
Atores envolvidos	Usuário do Sistema
Pré-condição	O usuário deve possuir um cadastro na ferramenta de visualização analítica; O usuário deve ter permissão de edição do *dashboard*.
Fluxo Principal	1. O usuário clica no título do componente a ser configurado; 2. O usuário clica em ‘*Edit’*; 3. O usuário acessa a guia ‘*Alert’*; 4. O usuário preenche nome, tempo de avaliação, período a ser avaliado, condições a serem atendidas, o limite (*threshold*), a mensagem de alerta; 5. O usuário seleciona o canal de comunicação a ser empregado (só terá um: Telegram); 6. O usuário salva as alterações.
Fluxos Alternativos	Item 2: O usuário não possui permissões para editar componentes do *dashboard*; 1. O sistema não permite prosseguir.
	Item 5: Não há nenhum canal de comunicação a ser selecionado. 1. O sistema não permite prosseguir; 2. O usuário deve reportar ao administrador, que fará a adição do canal de comunicação.
Pós-condição	Disparo de notificações configurado para um componente do *dashboard*.

Fonte: Autoria própria

4.4 DEFINIÇÃO DA ARQUITETURA

Como previamente definido, a solução proposta faz o uso de três ferramentas de forma integrada: Collectd, InfluxDB e Grafana. A Figura 25 exibe uma visão geral dos componentes que vieram a compor a arquitetura, bem como a forma como foi feita a implantação. O número de agentes ilustrado na figura não necessariamente reflete o que foi implementado.

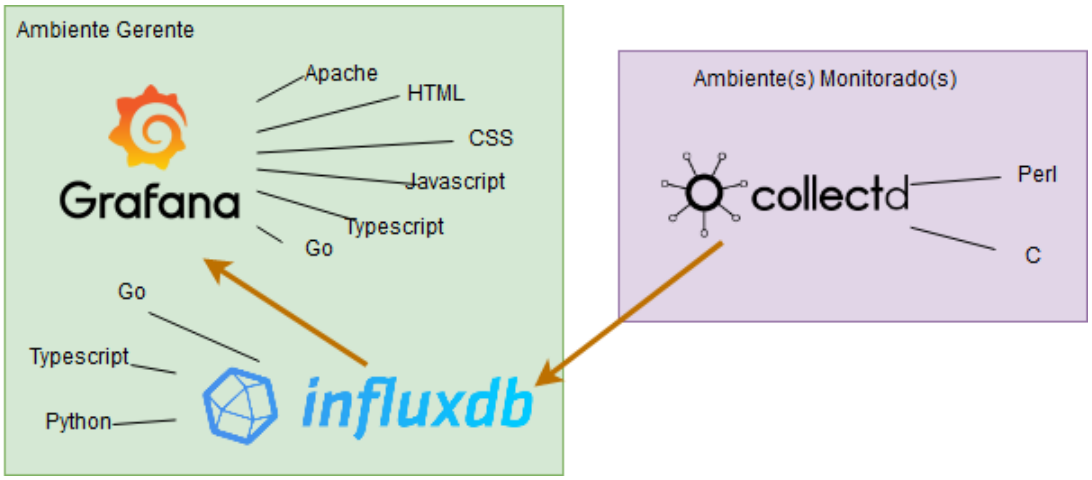
Figura 25 - Diagrama de componentes da arquitetura



Fonte: Autoria própria

A Figura 26 exibe, além das ferramentas a serem utilizadas, as tecnologias que foram empregadas para o desenvolvimento de cada *software* a ser integrado.

Figura 26 - Tecnologias utilizadas em cada ferramenta



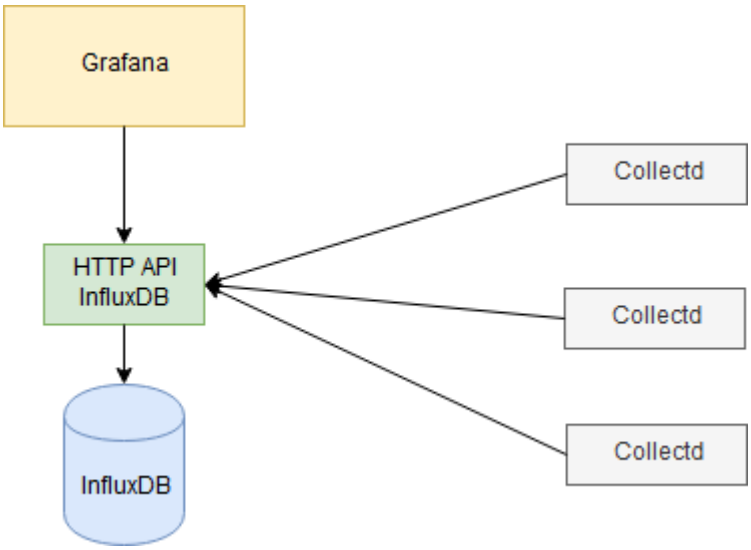
Fonte: Autoria própria

Nos elementos computacionais a serem monitorados, foi feita a instalação do Collectd, que foi executado de forma contínua como um *daemon*, ou seja, um serviço em *background*. Também foi instalado o *plugin* collectd-disk, que garantiu ao agente a capacidade de coletar dados de discos rígidos. A granularidade a ser empregada inicialmente foi de 5 segundos, e após

a validação da solução, tal intervalo foi mantido visto que apresentou excelentes resultados no que tange a performance no armazenamento e consulta de métricas.

Via rede e através de pacotes UDP, as métricas coletadas foram enviadas constantemente ao banco de dados de séries temporais, o InfluxDB, o qual inseria os pontos de tempo. O mesmo manteve em execução uma API HTTP na porta 8086, a qual possui *endpoints* para a recepção de dados para escrita e leitura. A Figura 27 ilustra a forma como a API HTTP do InfluxDB age como uma interface de fácil integração entre o banco de dados e ferramentas externas.

Figura 27 - API HTTP do InfluxDB



Fonte: Autoria própria

Diferentemente de bancos de dados relacionais, o banco de dados de séries temporais InfluxDB não possui relacionamentos, e as tabelas são denominadas ‘mensurações’ (INFLUXDATA, 2019). Foi feito o uso de 14 mensurações para o armazenamento de métricas, considerando a saída dos agentes de coleta e as métricas previamente selecionadas.

Visto que o agrupamento de informações se dá pela natureza dos mesmos, as métricas de diferentes agentes ficam armazenadas juntas, diferenciando-se por uma *tag* (ou índice) (INFLUXDATA, 2019). O Collectd utiliza a palavra “*host*” para esta *tag*, e foi possível escolher entre atribuir um nome à máquina monitorada de forma manual ou automática, através do arquivo de configuração do agente (COLLECTD, 2014). O Quadro 16 demonstra as mensurações a serem utilizadas e seus propósitos.

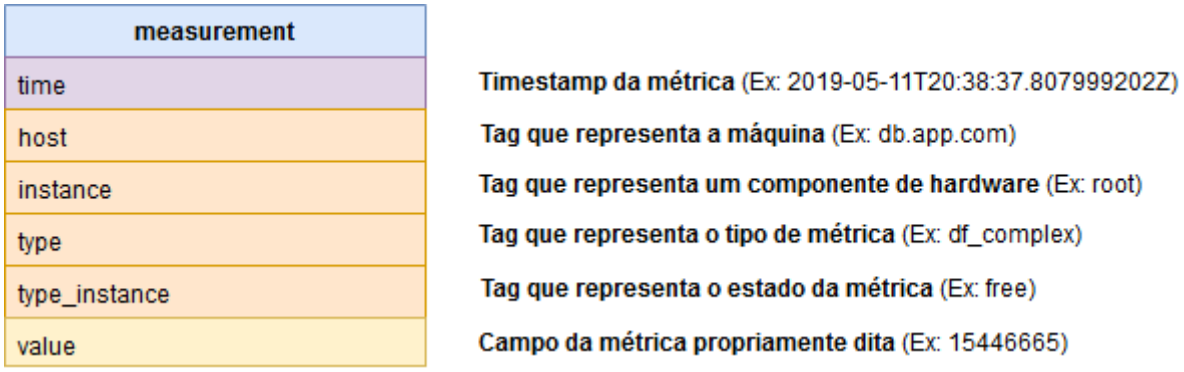
Quadro 16 - Mensurações a serem armazenadas no banco de dados

Mensuração	**Propósito de Armazenamento**
cpu_value	Tempos de ocupação de CPU por tipo de tarefa
df_value	Capacidade livre e ocupada por partição
disk_io_time	Tempo gasto em IO
disk_read	Octetos, tempo, operações e *merges* de leitura de disco
disk_weighted_io_time	Tempo em completar IO e possível *backlog* acumulado
disk_write	Octetos, tempo, operações e *merges* de escrita de disco
interface_rx	Octetos, pacotes e erros de entrada de rede em cada interface
interface_tx	Octetos, pacotes e erros de saída de rede em cada interface
load_longterm	Porcentagem de uso de CPU nos últimos 15 minutos
load_midterm	Porcentagem de uso de CPU nos últimos 5 minutos
load_shortterm	Porcentagem de uso de CPU no último 1 minuto
memory_value	Memória RAM livre, usada, em *cache* ou *buffer*
swap_value	Memória SWAP livre, usada, em *cache* ou *buffer*
uptime_value	Tempo em operação ininterrupta da máquina

Fonte: Autoria própria

A estrutura de cada mensuração é relativamente simples, sendo composta por um *timestamp*, *tags* e campos. A Figura 28 ilustra o modelo empregado, de uma forma geral.

Figura 28 - Modelo de mensuração utilizado no InfluxDB



Fonte: Autoria própria

Em algumas mensurações, nem todas as *tags* foram empregadas, devido à estrutura de dados que não exige maior especificação.

No mesmo servidor onde o InfluxDB foi instalado e executado para a recepção de métricas, a ferramenta de visualização analítica, o Grafana, foi executada. Conectando-se à mesma API HTTP que os agentes, a ferramenta faz requisições ao InfluxDB em um intervalo definido, o que permite a obtenção de dados para a renderização de componentes de *dashboard*.

4.5 VALIDAÇÃO DA SOLUÇÃO

A etapa de validação da solução foi iniciada a partir do momento em que o banco de dados já estava sendo abastecido com métricas de agentes, e a integração entre as ferramentas no ambiente gerente já havia sido finalizada. Utilizou-se uma amostra de 30 dias de métricas coletadas de 5 agentes, de forma a simular da forma mais real possível a implementação.

Para a validação, a solução proposta foi implantada na empresa AgênciaNet, de Flores da Cunha. Trata-se de uma empresa de pequeno porte (8 colaboradores), cujos serviços englobam o fornecimento de ferramentas de *marketing*, o desenvolvimento de *websites*, sistemas para *internet*, fornecimento de *e-mails*, e mais recentemente, o fornecimento de uma plataforma de anúncios de veículos (Ache Veículos) para revendedores do ramo. A empresa possui uma filial em Caxias do Sul, onde a infraestrutura de TI se encontra.

A empresa possui uma estrutura funcional simples, com setores administrativo, financeiro, TI, suporte, vendas e desenvolvimento. O setor de TI agiu como facilitador para a implantação da solução, uma vez que é o responsável pelo gerenciamento de infraestrutura de tecnologia que suporta as operações da empresa. Como descrito no capítulo 4.1, a implantação ocorreu em um servidor *on premise* da empresa, localizado na filial, na cidade de Caxias do Sul.

Para testar o correto funcionamento das ferramentas, foram executados os casos de teste abaixo listados.

- T-001: *Login* na ferramenta Grafana;
- T-002: Cadastro de usuário na ferramenta Grafana;
- T-003: Adição de novas máquinas monitoradas;
- T-004: Troca de visualização de máquina monitorada;
- T-005: Seleção de intervalo de tempo no *dashboard* de visualização das métricas;
- T-006: Aumento de detalhamento das métricas nos componentes do *dashboard*;
- T-007: Configuração de alertas em um painel de métricas do *dashboard*;
- T-008: Recebimento de alertas via grupo no mensageiro Telegram.

Ainda, foram coletadas informações importantes para atestar a viabilidade do projeto, visto que se relacionam diretamente ao consumo de recursos computacionais. São elas:

- M-001: Tráfego enviado ao ambiente gerente por cada agente em cada requisição;

- M-002: Espaço ocupado pelas métricas no banco de dados de séries temporais;
- M-003: *Delay* ao consultar métricas em diferentes períodos de tempo.

O projeto pode ser considerado válido se todos os testes acima citados tiverem resultados satisfatórios, e se as informações coletadas estiverem dentro das limitações impostas pelo ambiente onde a solução será implantada, ou se estiverem dentro de níveis aceitáveis (no caso de performance e *delays*). Foi também avaliada a conformidade com os requisitos não funcionais.

Ainda, para a validação da proposta, foram analisados os resultados obtidos na implantação da solução na empresa acima citada. Para isso, as métricas coletadas foram comparadas com os dados dos elementos monitorados, no que tange os picos de utilização, estatísticas de *uptime* e a utilização de recursos computacionais em momentos de execução de tarefas agendadas, por exemplo.

4.6 INTERFACES GRÁFICAS

Para ilustrar o planejamento das interfaces gráficas de usuário, foram desenvolvidos protótipos que demonstram as principais funcionalidades.

Visto que o *software* de visualização analítica possui uma interface estática, a qual não mudou na implementação do projeto, alguns dos protótipos são muito semelhantes à versão final. Também há protótipos que não representam a versão final, porém são necessários para representar a ideia da integração e o funcionamento da ferramenta.

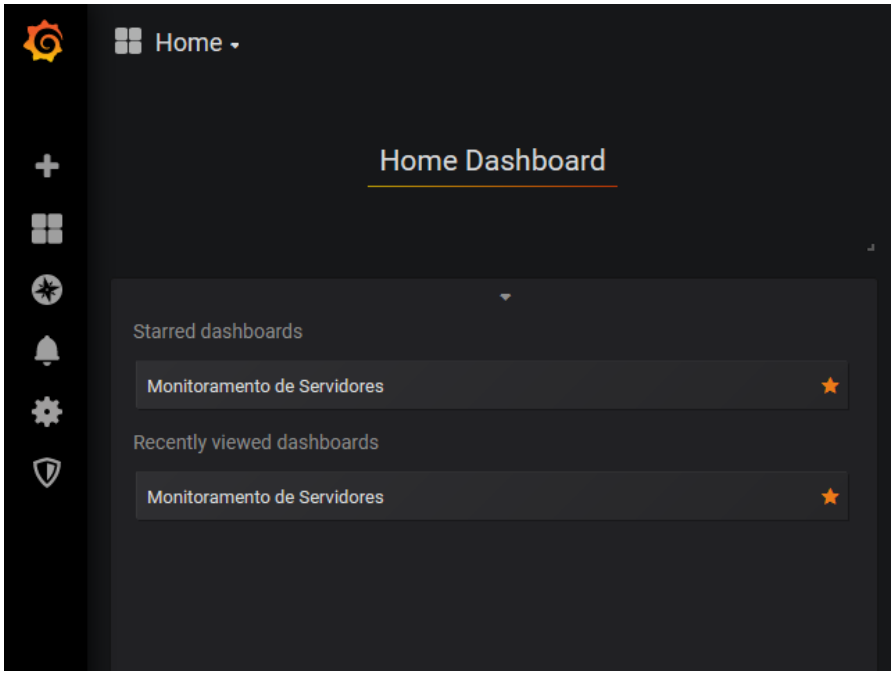
A Figura 29 demonstra a tela de *login* da ferramenta de visualização analítica, onde as métricas serão visualizadas. Há três componentes básicos: o campo de usuário (ou *e-mail*), o campo de senha, e o botão de *login*. Para efetuar o *login*, ambos os campos devem ser preenchidos corretamente.

Figura 29 - Tela de login do Grafana

Fonte: Autoria própria

A Figura 30 ilustra a tela de escolha de *dashboards* criados. O Grafana permite manter um número ilimitado de *dashboards*, mas para o propósito deste projeto, foram criados apenas o suficiente para a validação da proposta de solução.

Figura 30 - Tela de escolha de dashboard



Fonte: Autoria própria

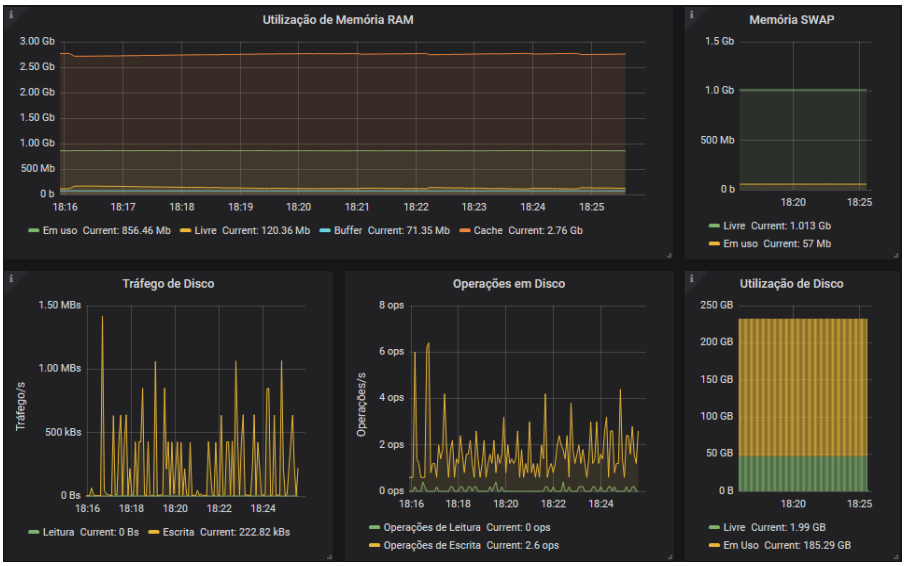
A Figura 31 ilustra a tela principal da aplicação: o *dashboard*, no qual ficam todos os componentes de exibição de métricas. É possível alterar o posicionamento e tamanho de cada componente adicionado, e é nesta tela onde se encontra os seletores de máquinas e de intervalos de tempo.

Figura 31 - Dashboard de monitoramento de métricas

Fonte: Autoria própria

A Figura 32 demonstra outros componentes que podem ser adicionados ao *dashboard*.

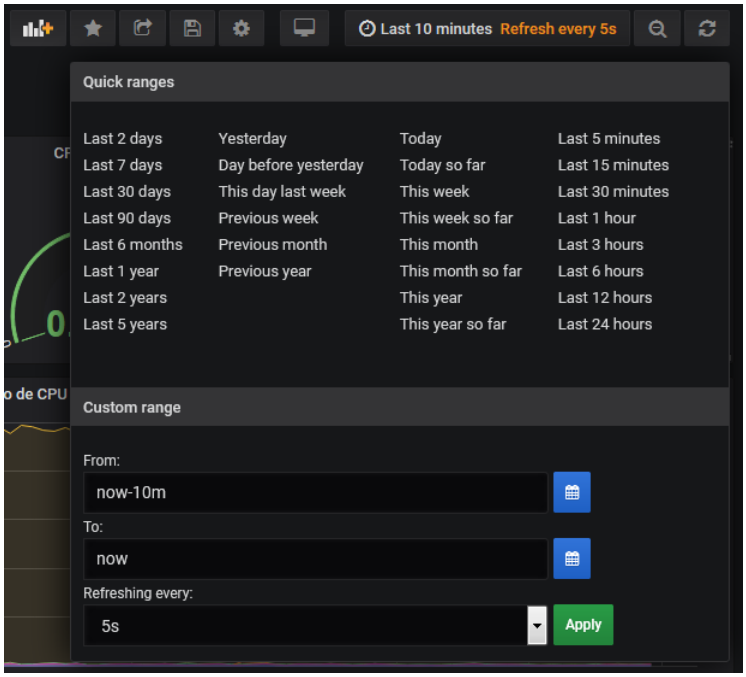
Figura 32 - Outros componentes de dashboard



Fonte: Autoria própria

Na Figura 33, pode-se ver de forma mais detalhada o seletor de intervalo de tempo, no *dashboard* de métricas. Ele é composto por intervalos pré-definidos, relativos ao *timestamp* corrente, porém há a liberdade de acessar um *timestamp* absoluto, especificando nos campos '*From*' e '*To*'. Ainda é possível definir um intervalo de atualização automática.

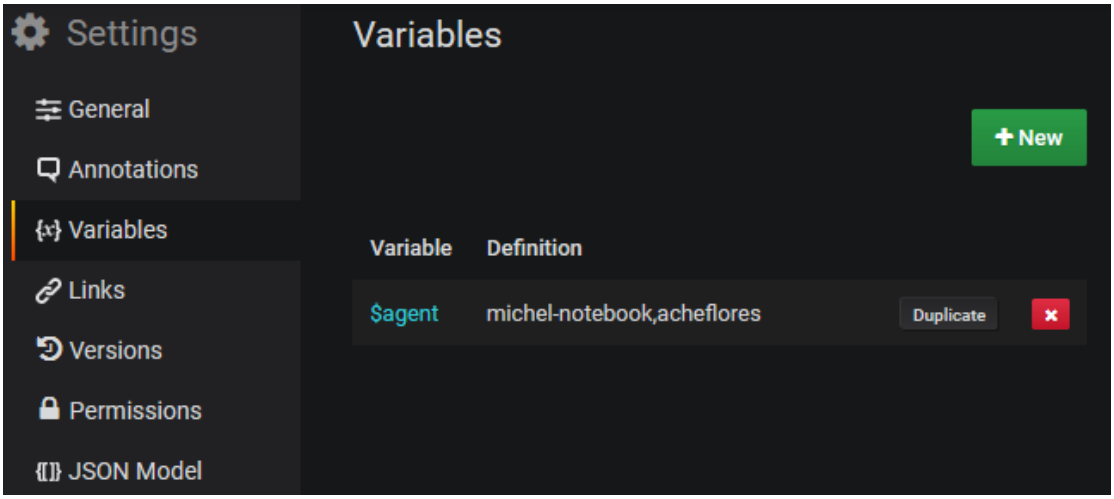
Figura 33 - Seletor de intervalos de tempo



Fonte: Autoria própria

A Figura 34 exibe a tela de configurações de *dashboard*, mais especificadamente a configuração de variáveis.

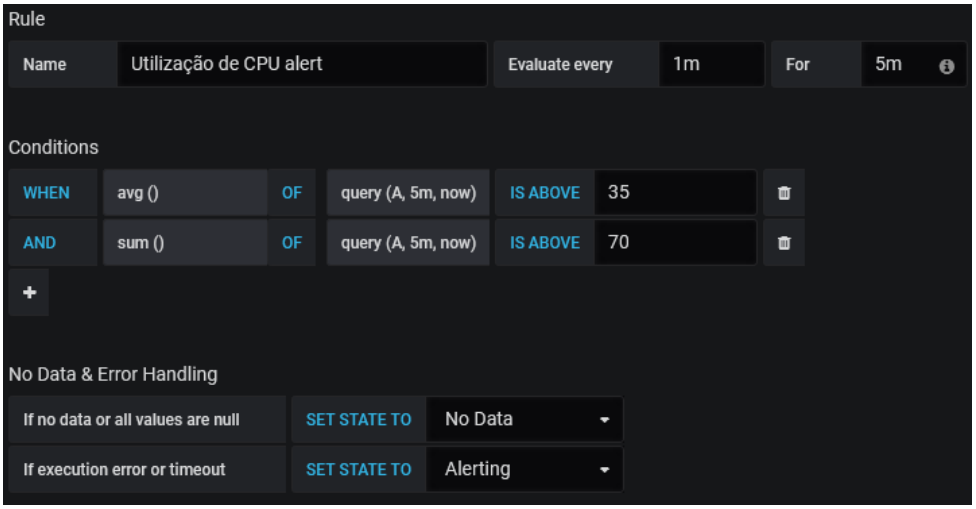
Figura 34 - Tela de configuração de variáveis



Fonte: Autoria própria

A Figura 35 exibe algumas das opções disponíveis no momento da configuração de alertas. Esta funcionalidade pode ser acessada em quaisquer componentes criados no *dashboard*.

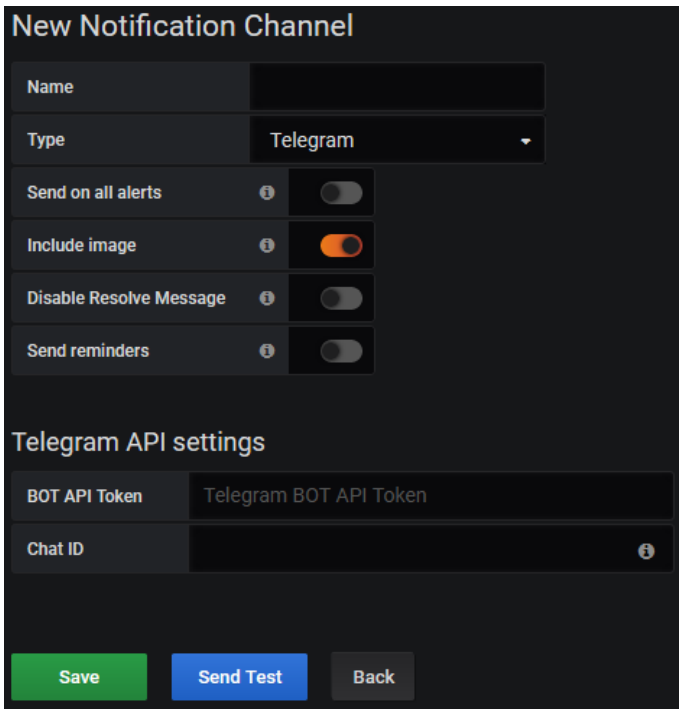
Figura 35 - Configuração de alertas



Fonte: Autoria própria

A Figura 36 exibe a tela disponível para a criação de canais de comunicação, ação a ser executada pelo usuário administrador do *software* de visualização analítica.

Figura 36 - Tela de configuração de canal de notificações



Fonte: Autoria própria

4.7 CONSIDERAÇÕES FINAIS

Neste capítulo, detalhou-se os aspectos técnicos da integração dos *softwares* selecionados, considerando as finalidades dos mesmos (documentadas na seção 3.1) e o contexto no qual o projeto será posto em prática. Foram definidas as especificações do ambiente que irá hospedar o banco de dados de séries temporais (InfluxDB), bem como o *software* de visualização analítica (Grafana), e os requisitos funcionais e não funcionais foram concebidos.

Com os requisitos, foi possível formular os casos de uso que irão servir como um guia para a implementação da integração, de forma que todas as funcionalidades vitais para atingir os objetivos do projeto estejam disponíveis e funcionais.

5 DESENVOLVIMENTO DA INTEGRAÇÃO

A integração das ferramentas *open-source* Collectd, InfluxDB e Grafana, selecionadas através da análise de critérios previamente selecionados, foi realizada com o objetivo de oferecer uma forma de monitorar métricas de baixo nível de servidores Linux em micro e pequenas organizações, fazendo uso de ferramentas gratuitas, infraestrutura própria e empregando procedimentos de fácil entendimento.

A integração foi planejada e desenvolvida através do transporte de dados entre as ferramentas. Considerando que, durante a etapa de seleção de ferramentas, a retrocompatibilidade e capacidade de disponibilização de dados através de integração foram critérios analisados, o desenvolvimento da integração não exigiu a programação de código-fonte com o intuito de coletar, transformar e enviar dados: as próprias ferramentas já possuem tais funcionalidades.

5.1 PREPARAÇÃO DO AMBIENTE

Para a execução do projeto, foram utilizados os seguintes *softwares* e sistema operacional na máquina de desenvolvimento:

- Windows 10;
- Putty 0.72;
- InfluxDB Studio v0.2.0 Beta 1.

Na máquina servidor, os *softwares* e sistema operacional utilizados foram:

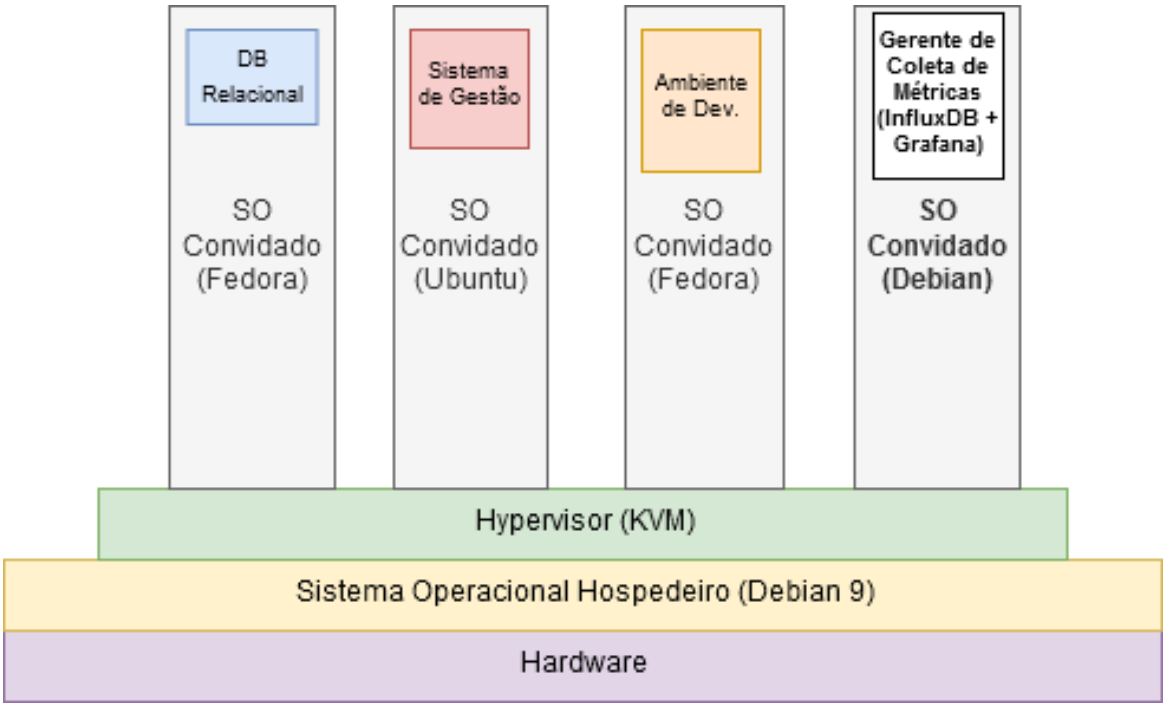
- Linux Debian 10 (Kernel 4.19.0-4);
- QEMU/KVM;
- InfluxDB 1.7.6;
- Grafana 6.2.5.

Nas máquinas clientes, cujas métricas foram coletadas, utilizou-se:

- Collectd 5.9.0;
- Collectd-disk 5.8.1.

Visto que o gerente da integração foi instalado em uma máquina *on premise* da organização, surgiu a necessidade de separar a solução implantada das outras aplicações, as quais são de importância estratégica para a organização. Fez-se uso de virtualização baseada em *kernel* utilizando o Hypervisor KVM e o emulador QEMU para criar um ambiente para a implantação, bem como evitar quaisquer problemas decorrentes de utilização de recursos. A Figura 37 demonstra a organização interna do ambiente.

Figura 37 - Estrutura do servidor hospedeiro



Fonte: Autoria própria

Para o acesso às máquinas (tanto gerente como clientes) foi empregado o protocolo SSH através do *software* Putty, que permitiu a criação e instalação de pares de chaves RSA de 4096 *bits*, facilitando o acesso recorrente para instalações e testes.

5.2 INSTALAÇÃO E PARAMETRIZAÇÃO DE FERRAMENTAS

Para a integração das ferramentas, foram feitas as instalações e realizadas customizações em relação aos parâmetros padrão. Alguns dos ajustes tiveram como objetivo definir a comunicação entre os elementos de rede, outros buscaram aumentar a segurança e performance, e também foi necessário definir a configuração de coleta.

5.2.1 Instalação e parametrização do InfluxDB

Primeiramente, foi feita a instalação do InfluxDB. Para isso, o repositório da aplicação foi adicionado, e foram executados os comandos de instalação característicos da distribuição Debian. A instalação seguiu o procedimento padrão documentado no site do desenvolvedor, que envolveu a utilização de privilégios de administrador, a liberação das portas 8086 (para a API HTTP) e 8088 (para chamadas RPC), e foi feita a customização do banco de dados de séries temporais. A Figura 38 demonstra as configurações aplicadas ao arquivo *influxdb.conf*.

Figura 38 - Alterações no arquivo influxdb.conf

```
[http]
enabled = true
bind-address = ":8486"
auth-enabled = true

[collectd]
enabled = true
bind-address = ":25826"
retention-policy = ""
database = collectd
typesdb = "/usr/local/share/collectd"
batch-size = 50
batch-pending = 10
batch-timeout = "5s"
read-buffer = 0
parse-multivalue-plugin = "split"
```

Fonte: Autoria própria

Foi ativada a autenticação e a API HTTP para recepção de dados. Ainda, foi ativado o módulo de ETL do Collectd (adicionando a biblioteca de tipos com o parâmetro '*typesdb*'), e definiu-se o nome do banco de dados como "*collectd*" para fácil identificação.

Considerando que para a validação da proposta foram utilizadas 5 máquinas com agentes coletores, foi empregado o valor de 50 pontos de tempo[5] para '*batch-size*', e de 10 pontos para '*batch-pending*', sendo que o '*timeout*' de 5 segundos força a gravação dos dados em disco se o lote não atingir o número de pontos configurado. Estes três parâmetros fazem com que o InfluxDB grave em disco os dados recebidos quase que imediatamente, evitando a

[5] Um ponto de tempo representa um registro único em um banco de dados de séries temporais, o qual possui quatro componentes: uma mensuração, uma tag, um campo e um *timestamp*.

falta de dados em tempo real para renderização e alertas. A partir do momento em que a integração passar a receber métricas de um número de coletores maior, faz-se necessário analisar a capacidade da máquina gerente em armazenar tais pontos de tempo em tempo hábil, e deve-se reavaliar tais parâmetros.

A diretiva '*parse-multivalue-plugin*' empregada com o valor "*split*" confere um comportamento fundamental para a integração: dados coletados de forma conjunta são armazenados em mensurações diferentes, simplificando o processo de consultas. Como a sintaxe do InfluxQL não possui funções de *parsing*, esta diretiva simplifica as consultas necessárias separando os dados de forma mais eficiente.

Visto que para o contexto proposto, estima-se que a retenção máxima necessária seja de três meses (considerado tempo suficiente para a identificação de quaisquer padrões de utilização de recursos e desempenho), foi estabelecida uma política de retenção de 90 dias, com um tamanho de *shard*[6] de 24 horas. Logo, métricas armazenadas que ultrapassarem os 90 dias de idade serão descartadas automaticamente, em blocos de 24 horas por vez. A Figura 39 exibe a consulta de criação da política de retenção, que passa a ser a política padrão do banco de dados "*collectd*".

Figura 39 - Consulta de criação da política de retenção

```
1  CREATE RETENTION POLICY "3months"
2  ON "collectd"
3  DURATION 90d
4  REPLICATION 1
5  SHARD DURATION 24h
6  DEFAULT
```

Fonte: Autoria própria

Por fim, o serviço da aplicação (*influxdb.service*) foi habilitado e inicializado, permitindo a conexão remota para execução de *queries*.

5.2.2 Instalação e parametrização do Collectd

A instalação do Collectd nas máquinas a serem monitoradas ocorreu através de procedimentos comuns de instalação via gerenciadores de pacotes (como YUM, APT, DNF ou

[6] Um shard contém um conjunto específico de pontos de tempo, e no InfluxDB, é representado por um arquivo TSM (Time-Series Merge) em disco.

pacman), sendo que não houve a necessidade de fazer a instalação direta por meio de download e compilação. Na configuração da ferramenta de coleta de dados (Collectd), foi necessária a definição de parâmetros. A Figura 40 exibe o conteúdo do arquivo de configuração *collectd.conf*.

Figura 40 - Arquivo de configuração *collectd.conf*

```
Hostname    "acheveiculos.com"
Interval     5

LoadPlugin syslog
LoadPlugin cpu
LoadPlugin df
LoadPlugin disk
LoadPlugin interface
LoadPlugin load
LoadPlugin memory
LoadPlugin network
LoadPlugin swap
LoadPlugin uptime
<Plugin df>
  IgnoreSelected true
</Plugin>

<Plugin disk>
  Disk "vda"
</Plugin>

<Plugin load>
</Plugin>

<Plugin memory>
</Plugin>

<Plugin network>
  <Server "metricas.agencianet.net.br" "25826">
    SecurityLevel None
    Username "admin"
    Password "mji.1992"
  </Server>
</Plugin>

<Plugin swap>
</Plugin>

Include "/etc/collectd.d"
```

Fonte: Autoria própria

Definiu-se um *hostname* que identifica a máquina monitorada e o intervalo de coletas. Para fim de validação da proposta de solução, foi definido o intervalo de 5 segundos entre coletas. Tal valor garante uma visualização detalhada o suficiente para uma análise gráfica, ao mesmo tempo que mantém a utilização de recursos pelo agente em níveis aceitáveis, e não consome um espaço em disco a ponto de impedir a implementação da solução no contexto proposto, como pode ser visto no capítulo 6.

Ainda, outros parâmetros ativados e configurados foram os *plugins* de coleta. Dentre eles: "df", responsável pela coleta de métricas de utilização de disco, com a diretiva "*IgnoreSelected'* indicando que devem ser consideradas todas as partições com permissão de leitura; "cpu" para coleta de métricas de processamento; "*load*" para coleta de métricas de *run-queue length*; "*interface*" para coleta de métricas referentes à interface de rede; "*memory*" e "swap" para métricas referentes à memória RAM e memória SWAP, respectivamente; "*uptime*" para a coleta do tempo desde o boot, e "*network*", sendo o plugin responsável pelo envio de métricas para o ambiente externo.

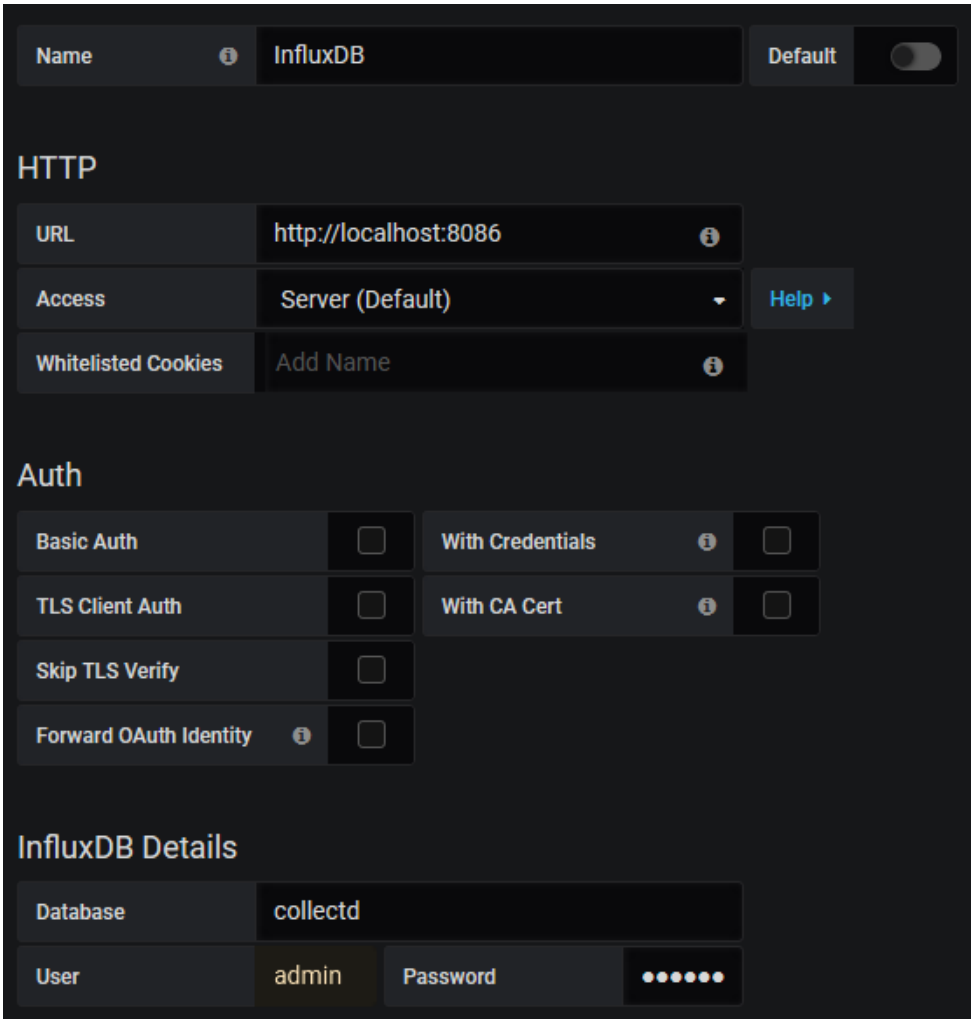
O plugin '*syslog*' foi ativado para fins de *debugging*, visto que ele registra no gerenciador de logs do sistema as mensagens referentes a possíveis falhas que possam vir a ocorrer com a coleta e/ou envio de métricas. Por fim, o serviço do agente (collectd.service) foi habilitado e inicializado.

5.2.3 Instalação e parametrização do Grafana

Para a instalação e configuração da ferramenta de visualização analítica (Grafana), os procedimentos apontados na documentação oficial do projeto foram seguidos. Fazendo uso de privilégios de administrador, foi adicionado o repositório da ferramenta, e através do comando '*apt-get*' (gerenciador de pacotes), foi feita a busca e instalação do pacote. Após a instalação, foi habilitado e iniciado o serviço da aplicação (*grafana-server*), e após a liberação da porta de rede (3000), todo o procedimento posterior pôde ser feito através da interface gráfica.

O único procedimento de configuração necessário antes da criação de *dashboard* foi a adição da fonte dos dados a ser empregada no projeto: configurou-se a integração com o banco de dados de séries temporais InfluxDB, através do preenchimento de dados como usuário, senha, IP e porta, como pode ser visto na Figura 41.

Figura 41 - Adição de fonte de dado no Grafana

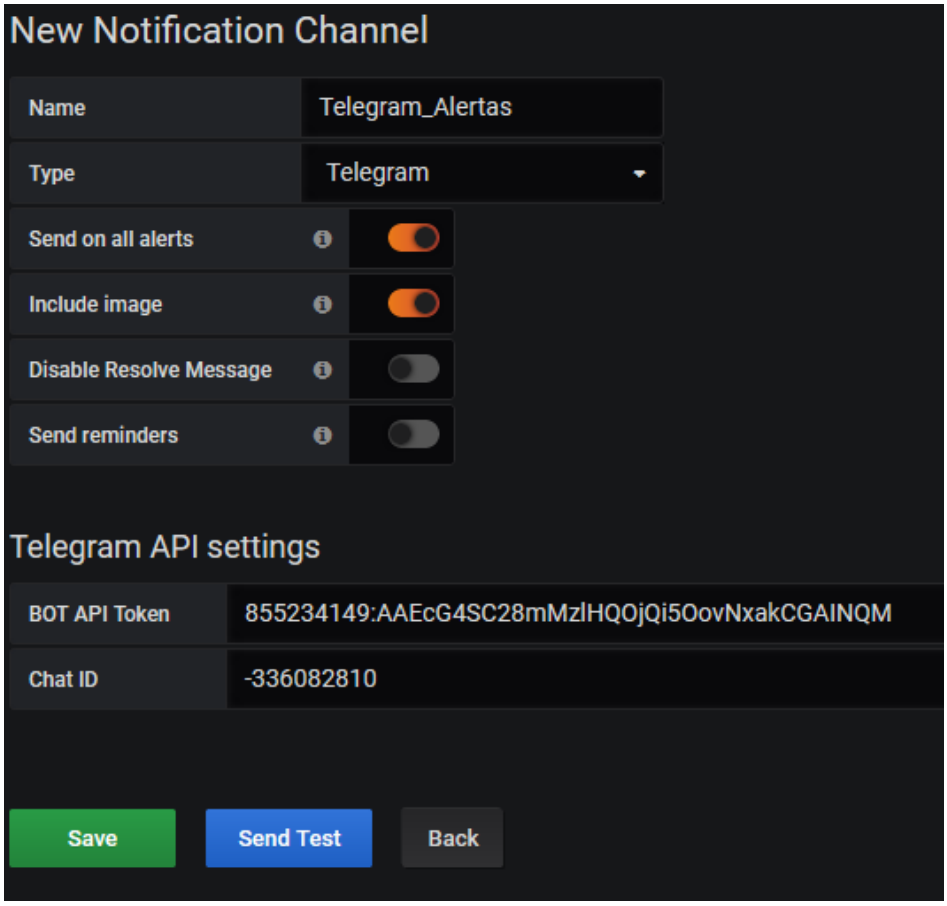


Fonte: Autoria própria

Para atender ao requisito de permitir o disparo de alertas de métricas via Telegram, ainda foi necessário criar um *bot*[7] para o mensageiro instantâneo, adicioná-lo em um grupo (o qual foi nomeado "Alertas Servidores" para fácil identificação), e configurar o *token* de autenticação do mesmo na ferramenta de visualização, além do ID do grupo de conversação. Para este procedimento, seguiu-se o passo-a-passo descrito na documentação do Telegram, e com os dados obtidos, o canal de comunicação foi adicionado. A Figura 42 exibe a tela de adição de canal de comunicação.

[7] Usuário especial cujo comportamento é definido através de algoritmos, que responde a comandos enviados e/ou disparam mensagens oriundas de aplicações de terceiros através de uma API.

Figura 42 - Tela de adição de canal de comunicação do Grafana



Fonte: Autoria própria

5.3 CONFIGURAÇÃO DOS PAINEIS DE CONTROLE

O funcionamento deste projeto de integração se dá por meio do transporte de dados coletados, os quais são enviados para um banco de dados de séries temporais, onde ficam armazenados para consultas pelo *software* de visualização analítica.

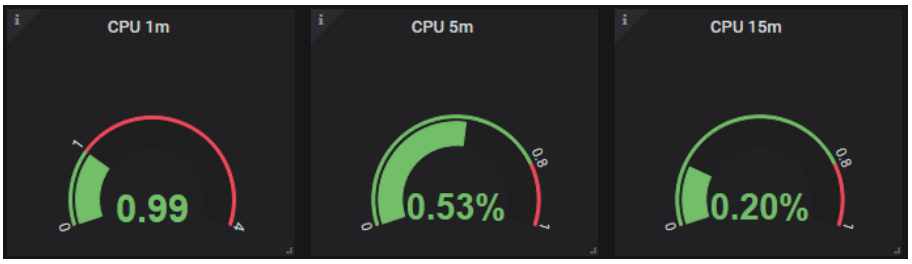
O *software* de visualização analítica faz uso dos dados armazenados para a renderização de elementos visuais (neste caso, gráficos e indicadores), além de executar tarefas previamente definidas (como envio de alertas).

Foi criado um *dashboard* contendo 11 elementos visuais, sendo que cada um relaciona-se com uma ou mais métricas coletadas. Os mesmos estão organizados em linhas, e possuem diferentes tamanhos de forma a facilitar a visualização das séries temporais.

5.3.1 Indicador de fila de execução

Os 3 indicadores exibem a carga de CPU, ou seja, o número de processos que estão consumindo tempo de processamento, somado também ao número de processos aguardando execução. Isto é comumente denominado *run-queue length*, ou tamanho da fila de execução. Cada indicador representa a média em relação a um período distinto: 1 minuto, 5 minutos e 15 minutos, como pode ser visto na Figura 43.

Figura 43 - Indicadores de *run-queue length*



Fonte: Autoria própria

Tais indicadores representam o último valor coletado dentro da faixa de tempo selecionada. A Figura 44 exibe as consultas InfluxQL correspondentes.

Figura 44 - Consultas de indicadores de fila de execução

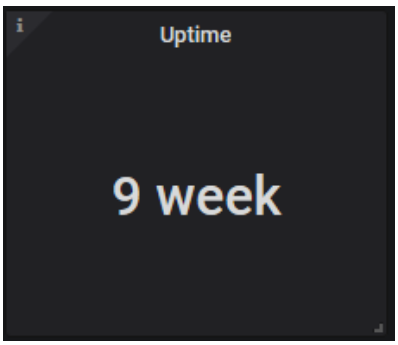
```
/* Consulta para o indicador de 1 minuto */
SELECT
  last("value")
FROM "load_shortterm"
WHERE ("host" = $host AND "type" = 'load') AND $timeFilter
GROUP BY time($__interval)
FILL(previous)

/* Consulta para o indicador de 5 minutos */
SELECT
  last("value")
FROM "load_midterm"
WHERE ("host" = $host AND "type" = 'load') AND $timeFilter
GROUP BY time($__interval)
FILL(previous)

/* Consulta para o indicador de 15 minutos */
SELECT
  last("value")
FROM "load_longterm"
WHERE ("host" = $host AND "type" = 'load') AND $timeFilter
GROUP BY time($__interval)
FILL(previous)
```

Fonte: Autoria própria

5.3.2 Indicador de tempo de boot

O único indicador textual é o que exibe o *uptime*. Trata-se do tempo que a máquina está em funcionamento desde seu último *boot*. Valores altos indicam que a máquina está em execução por mais tempo, e a unidade de medida é convertida automaticamente para dias, semanas, meses ou anos para facilitar a leitura. A Figura 45 exibe o elemento.

Figura 45 - Indicador de *uptime*



Fonte: Autoria própria

Para a renderização deste indicador, efetua-se uma consulta buscando o último valor compreendido entre as faixas de tempo selecionadas de forma semelhante ao indicador de fila de execução, como pode ser visto na Figura 46.

Figura 46 - Consulta de *uptime*

```
1  /* Consulta para o indicador de uptime */
2  SELECT
3    last("value")
4  FROM "uptime_value"
5  WHERE ("host" = $host AND "type" = 'uptime') AND $timeFilter
6  GROUP BY time($__interval)
7  FILL(previous)
```

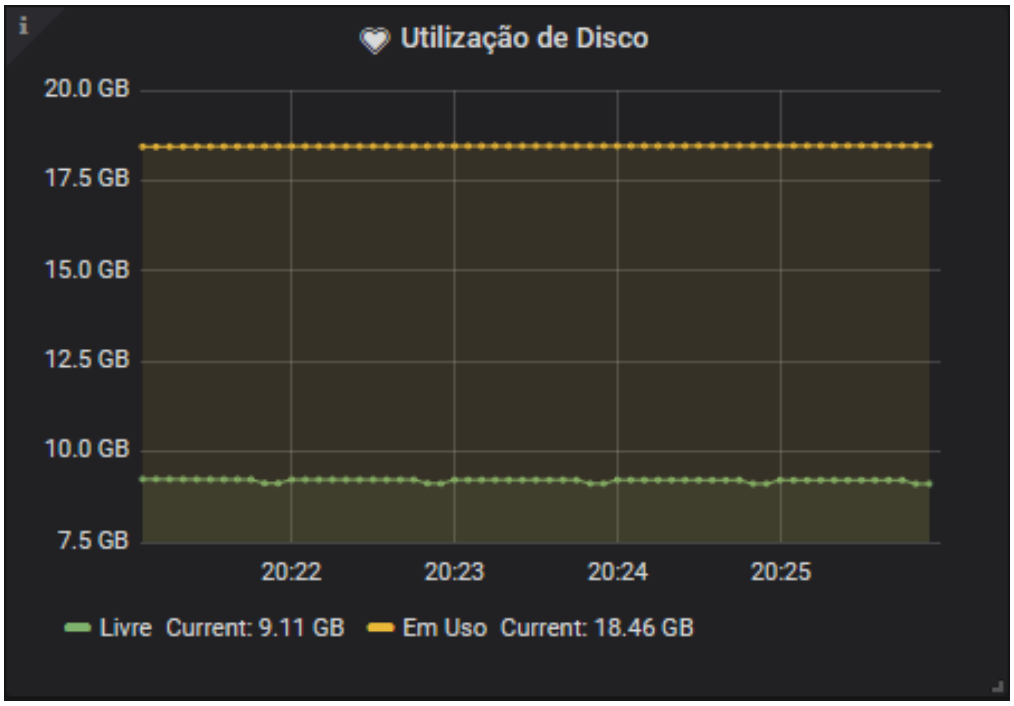
Fonte: Autoria própria

5.3.3 Gráfico de utilização de disco

O gráfico de linhas demonstra a utilização de disco fazendo uso de duas métricas coletadas: espaço livre, e espaço ocupado, cada uma representada por uma cor nas barras empilhadas, como pode ser visto na Figura 47. Visto que diferentes máquinas monitoradas

podem possuir diferentes partições e esquemas de armazenamento, coleta-se métricas de todas as partições existentes e montadas no sistema operacional.

Figura 47 - Gráfico de utilização de disco



Fonte: Autoria própria

Para a renderização do gráfico, efetua-se a soma do espaço livre e ocupado de todas as partições, como pode ser visto na Figura 48.

Figura 48 - Consultas para métricas de utilização de disco

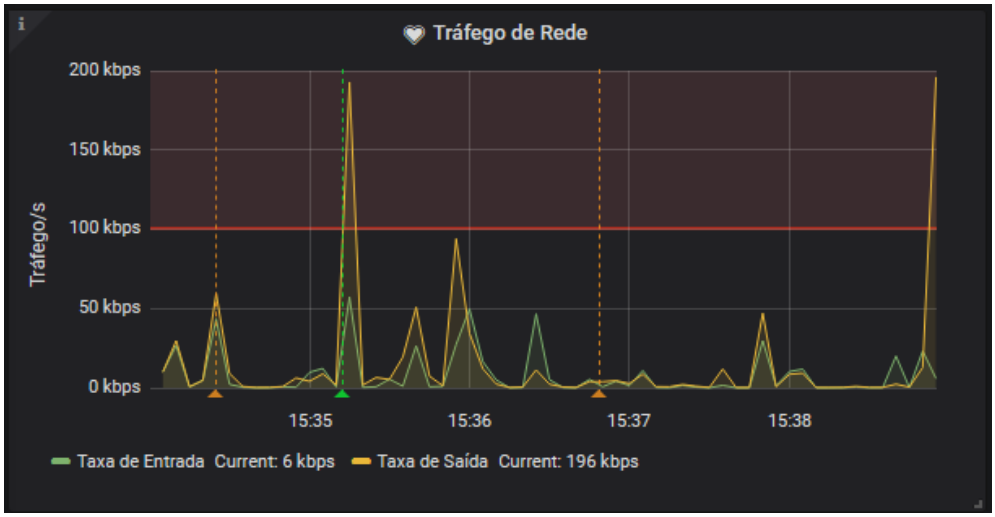
```
/* Consulta para a métrica de espaço livre */
SELECT
  SUM("value")
FROM "df_value"
WHERE ("host" = $host AND "type_instance" = 'free') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)

/* Consulta para a métrica de espaço ocupado */
SELECT
  SUM("value")
FROM "df_value"
WHERE ("host" = $host AND "type_instance" = 'used') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)
```

Fonte: Autoria própria

5.3.4 Gráfico de tráfego de rede

O gráfico do tráfego de rede faz uso de duas métricas coletadas: o número de octetos (*bytes*) enviados e recebidos. O elemento está ilustrado na Figura 49.

Figura 49 - Gráfico de tráfego de rede



Fonte: Autoria própria

Através de cálculos de derivada e divisão, converte-se os valores cumulativos coletados e pode-se obter a taxa de entrada e saída de dados por segundo, independente da frequência de atualização do gráfico. Também foi necessário empregar o recurso do Grafana que efetua a conversão de unidades. As *queries* correspondentes podem ser vistas na Figura 50.

Figura 50 - Consultas do gráfico de tráfego de rede

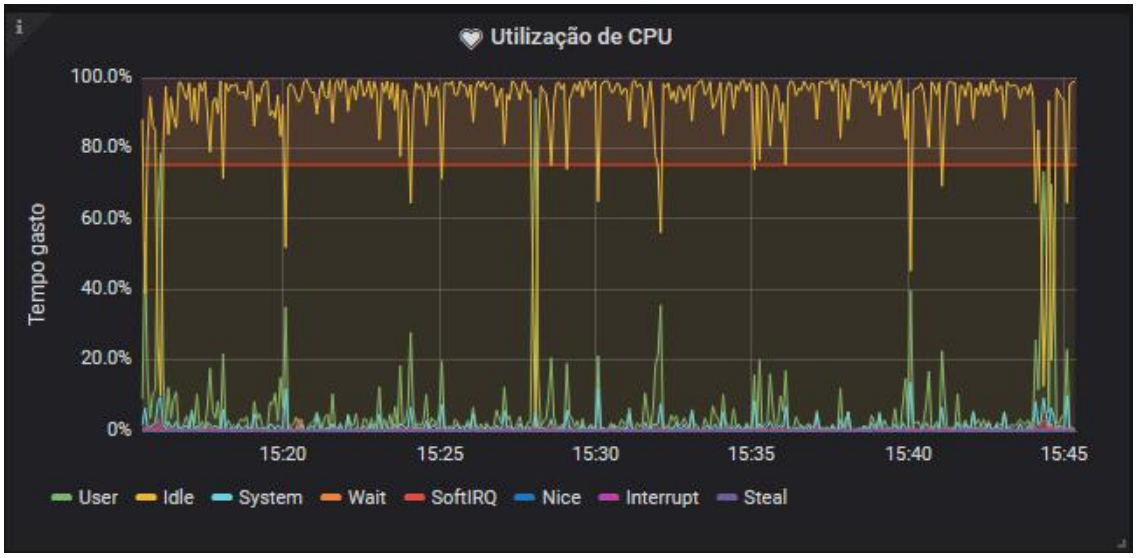
```
/* Consulta para a métrica de entrada de dados */
SELECT
  DERIVATIVE(MEAN("value"), 1s) / 1024
FROM "interface_rx"
WHERE ("host" = $host AND "type" = 'if_octets' AND "instance" = 'ens3') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)

/* Consulta para a métrica de saída de dados */
SELECT
  DERIVATIVE(MEAN("value"), 1s) / 1024
FROM "interface_tx"
WHERE ("host" = $host AND "type" = 'if_octets' AND "instance" = 'ens3') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)
```

Fonte: Autoria própria

5.3.5 Gráfico de utilização de CPU

O gráfico de utilização de CPU faz uso de 8 métricas coletadas para renderizar uma representação visual do tempo gasto pelo processador da máquina em executar determinadas tarefas. Os contextos de execução são *User* (tarefas executadas pelo usuário), *Idle* (tempo livre), *System* (tarefas executadas pelo sistema), *Wait* (tempo gasto na fila de execução, aguardando), *SoftIRQ* (tempo em interrupção por *hardware*), Nice (tempo gasto em tarefas de baixa prioridade), *Interrupt* (o mesmo que SoftIRQ), e *Steal* (tempo roubado por um possível hypervisor para uso de sistema inquilino). A Figura 51 ilustra o elemento.

Figura 51 - Gráfico de utilização de CPU



Fonte: Autoria própria

Visto que as máquinas monitoradas podem conter um número diferente de núcleos ou *threads*, o cálculo efetua uma média de tempo para cada estado, seguido de derivada não negativa (taxa de mudança entre os valores subsequentes), devido ao fato que o valor coletado é cumulativo, como pode ser visto na Figura 52.

Figura 52 - Consultas de métricas para o gráfico de utilização de CPU

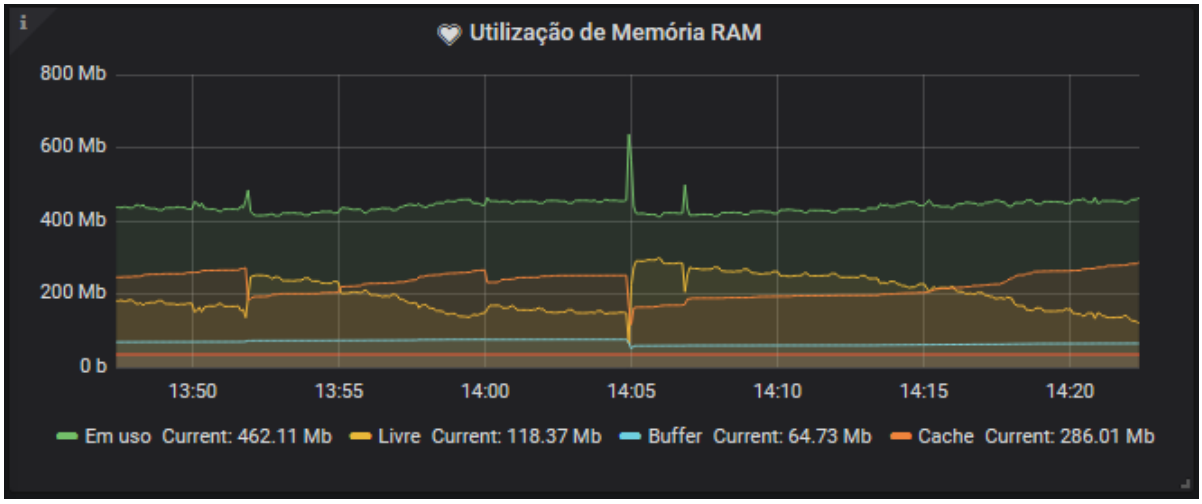
```
SELECT
  NON_NEGATIVE_DERIVATIVE(MEAN("value"), 1s) FROM "cpu_value"
WHERE ("host" = $host AND "type_instance" = 'user') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)

SELECT
  NON_NEGATIVE_DERIVATIVE(MEAN("value"), 1s) FROM "cpu_value"
WHERE ("host" = $host AND "type_instance" = 'idle') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)

SELECT
  NON_NEGATIVE_DERIVATIVE(MEAN("value"), 1s) FROM "cpu_value"
WHERE ("host" = $host AND "type_instance" = 'system') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)

SELECT
  NON_NEGATIVE_DERIVATIVE(MEAN("value"), 1s) FROM "cpu_value"
WHERE ("host" = $host AND "type_instance" = 'wait') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)

SELECT
  NON_NEGATIVE_DERIVATIVE(MEAN("value"), 1s) FROM "cpu_value"
WHERE ("host" = $host AND "type_instance" = 'softirq') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)

SELECT
  NON_NEGATIVE_DERIVATIVE(MEAN("value"), 1s) FROM "cpu_value"
WHERE ("host" = $host AND "type_instance" = 'nice') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)

SELECT
  NON_NEGATIVE_DERIVATIVE(MEAN("value"), 1s) FROM "cpu_value"
WHERE ("host" = $host AND "type_instance" = 'interrupt') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)

SELECT
  NON_NEGATIVE_DERIVATIVE(MEAN("value"), 1s) FROM "cpu_value"
WHERE ("host" = $host AND "type_instance" = 'steal') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)
```

Fonte: Autoria própria

5.3.6 Gráfico de utilização de memória RAM

O gráfico de memória RAM faz uso de 4 métricas coletadas para exibir a quantidade de memória livre, memória em uso, em *buffer* e em *cache*. Considerando que os valores coletados estão em *bits*, são feitos cálculos para a conversão em *megabytes*, e cada linha horizontal corresponde a uma das métricas coletadas, como ilustrado na Figura 53.

Figura 53 - Gráfico de utilização de memória RAM



Fonte: Autoria própria

A coleta das 4 métricas utilizadas se dá por meio das consultas ilustradas na Figura 54.

Figura 54 - Consultas para o gráfico de utilização de memória RAM

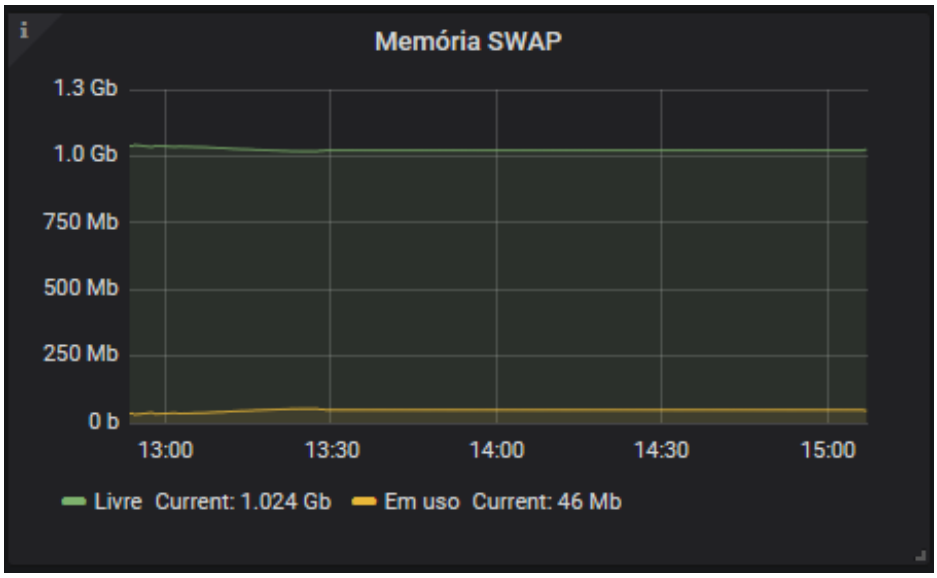
```
SELECT
  mean("value")
FROM "memory_value" WHERE ("host" = $host AND "type" = 'memory' AND "type_instance" = 'used') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)

SELECT
  mean("value")
FROM "memory_value" WHERE ("host" = $host AND "type" = 'memory' AND "type_instance" = 'free') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)

SELECT
  mean("value")
FROM "memory_value" WHERE ("host" = $host AND "type" = 'memory' AND "type_instance" = 'buffered') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)

SELECT
  mean("value")
FROM "memory_value" WHERE ("host" = $host AND "type" = 'memory' AND "type_instance" = 'cached') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)
```

Fonte: Autoria própria

5.3.7 Gráfico de utilização de memória SWAP

O gráfico de memória SWAP exibe as métricas de utilização de memória virtual (paginação). Embora tais métricas sejam coletadas em todas as máquinas monitoradas, há de se considerar que tal memória não é utilizada em todos os casos, fazendo com que o gráfico fique sem métricas renderizadas. Cada linha indica uma das métricas relacionadas: memória livre e memória ocupada, como indicado na Figura 55.

Figura 55 - Gráfico de utilização de memória SWAP



Fonte: Autoria própria

As consultas utilizadas para a renderização deste gráfico estão na Figura 56.

Figura 56 - Consultas para o gráfico de utilização de memória SWAP

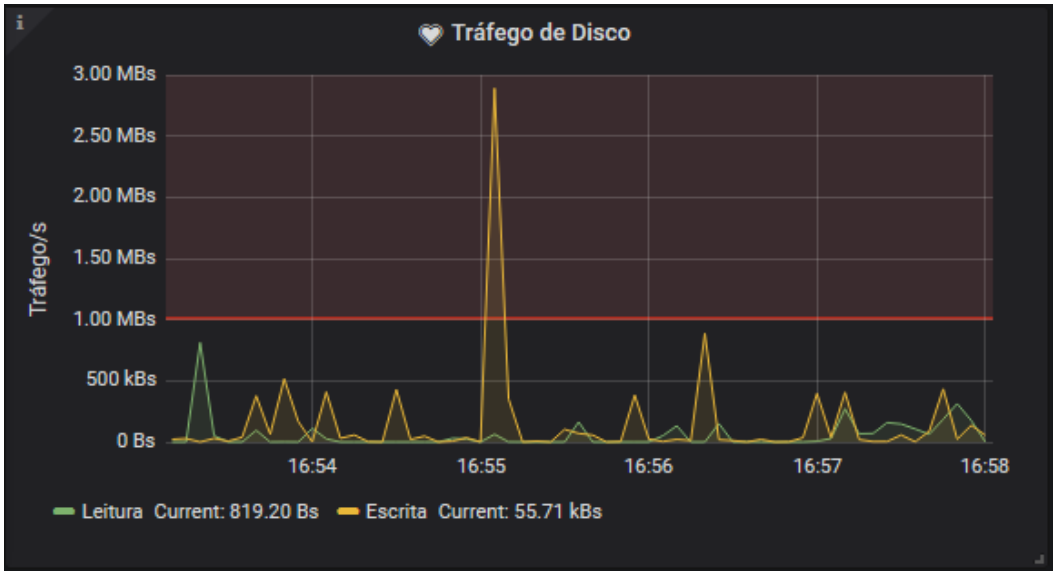
```
SELECT
  MEAN("value")
FROM "swap_value"
WHERE ("host" = $host AND "type" = 'swap' AND "type_instance" = 'free') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)

SELECT
  MEAN("value")
FROM "swap_value"
WHERE ("host" = $host AND "type" = 'swap' AND "type_instance" = 'used') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)
```

Fonte: Autoria própria

5.3.8 Gráfico de tráfego de disco

O tráfego de disco faz uso de duas métricas coletadas: o número de octetos gravados e lidos. A Figura 57 demonstra o gráfico renderizado.

Figura 57 - Gráfico de tráfego de disco



Fonte: Autoria própria

Para renderizar tal gráfico, efetua-se a média dos valores no intervalo de tempo selecionado, e aplica-se uma função derivada não negativa. O próprio Grafana se encarrega de converter os valores para *kilobytes* por segundo. A Figura 58 exibe as consultas InfluxQL aplicadas.

Figura 58 - Consultas para o gráfico de tráfego de disco

```
SELECT
  NON_NEGATIVE_DERIVATIVE(MEAN("VALUE"), 1s)
FROM "disk_read" WHERE ("host" = $host AND "type" = 'disk_octets') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)

SELECT
  NON_NEGATIVE_DERIVATIVE(MEAN("VALUE"), 1s)
FROM "disk_write" WHERE ("host" = $host AND "type" = 'disk_octets') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)
```

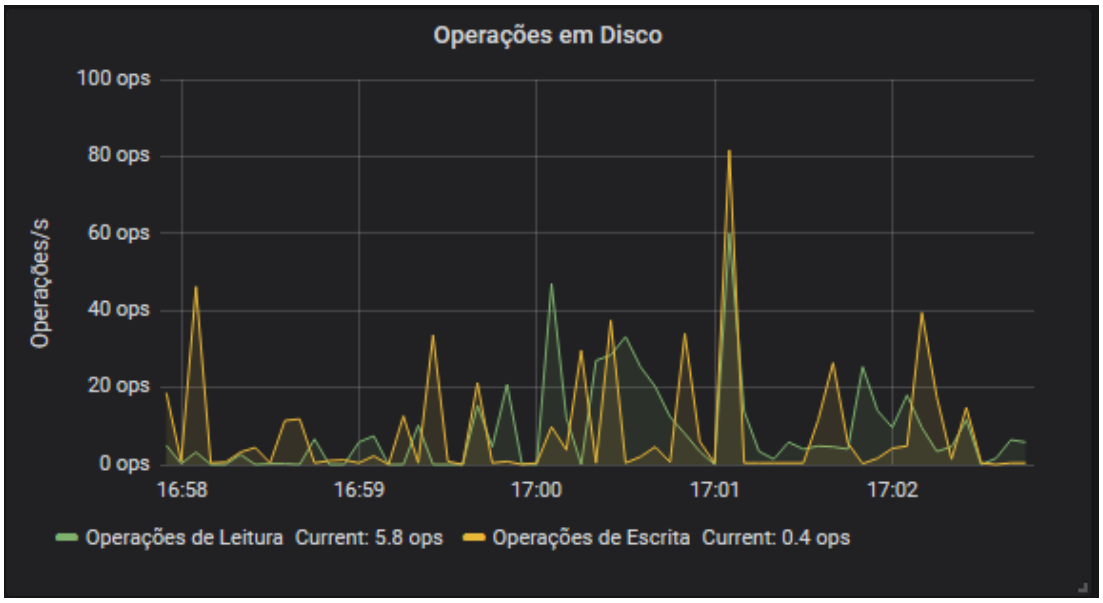
Fonte: Autoria própria

5.3.9 Gráfico de operações em disco

O gráfico de operações em disco exibe o número de operações de leitura e de escrita, por segundo. É importante considerar que o *plugin* de coleta considera não apenas as operações em execução como também as operações pendentes (na fila de espera), tanto para leitura como para escrita. Para a renderização, também são aplicados cálculos de média e derivada de valores, visto que os valores coletados são cumulativos e em *byte*.

A Figura 59 demonstra o gráfico.

Figura 59 - Gráfico de operações em disco



Fonte: Autoria própria

A Figura 60 exibe as consultas InfluxQL utilizadas.

Figura 60 - Consultas para o gráfico de operações em disco

```
SELECT
  NON_NEGATIVE_DERIVATIVE(MEAN("VALUE"), 1s)
FROM "disk_read" WHERE ("host" = $host AND "type" = 'disk_ops') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)

SELECT
  NON_NEGATIVE_DERIVATIVE(MEAN("VALUE"), 1s)
FROM "disk_write" WHERE ("host" = $host AND "type" = 'disk_ops') AND $timeFilter
GROUP BY time($__interval)
FILL(NULL)
```

Fonte: Autoria própria

5.4 FLUXO DE COMUNICAÇÃO ENTRE FERRAMENTAS

Após a implementação da solução, o fluxo de comunicação planejado na formulação da proposta de solução pôde ser empregado em uma situação real com sucesso. Feita a coleta das métricas pelo agente coletor, são enviados pacotes UDP para a máquina gerente. Considerando o tamanho máximo padrão de pacotes e as métricas selecionadas, fazem-se necessários três pacotes a cada 5 segundos (intervalo de coleta definido). Há de se considerar que, em uma situação onde são selecionadas mais métricas, naturalmente serão enviados um número maior de pacotes UDP por intervalo definido.

Estes pacotes possuem os dados de autenticação, identificação da máquina, o *timestamp* das métricas, e as métricas propriamente ditas. A Figura 61 exibe um *dump* da saída da interface de rede de uma das máquinas monitoradas.

Figura 61 - Saída do comando *tcpdump* de máquina monitorada

```
[root@michel mjisoton]# tcpdump udp -u port 25826
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlp2s0, link-type EN10MB (Ethernet), capture size 262144 bytes
21:33:23.180048 IP michel.notebook.44521 > metricas.agencianet.net.br.25826: UDP, length 1323
21:33:23.180548 IP michel.notebook.44521 > metricas.agencianet.net.br.25826: UDP, length 1334
21:33:23.181357 IP michel.notebook.44521 > metricas.agencianet.net.br.25826: UDP, length 1307
21:33:28.179894 IP michel.notebook.44521 > metricas.agencianet.net.br.25826: UDP, length 1331
21:33:28.180502 IP michel.notebook.44521 > metricas.agencianet.net.br.25826: UDP, length 1316
21:33:28.181481 IP michel.notebook.44521 > metricas.agencianet.net.br.25826: UDP, length 1339
21:33:33.180273 IP michel.notebook.44521 > metricas.agencianet.net.br.25826: UDP, length 1343
21:33:33.180693 IP michel.notebook.44521 > metricas.agencianet.net.br.25826: UDP, length 1299
21:33:33.181545 IP michel.notebook.44521 > metricas.agencianet.net.br.25826: UDP, length 1284
21:33:38.180167 IP michel.notebook.44521 > metricas.agencianet.net.br.25826: UDP, length 1339
21:33:38.180627 IP michel.notebook.44521 > metricas.agencianet.net.br.25826: UDP, length 1340
21:33:38.181652 IP michel.notebook.44521 > metricas.agencianet.net.br.25826: UDP, length 1314
```

Fonte: Autoria própria

Estes dados são recebidos pela API HTTP do banco de dados, e através de um módulo de ETL que faz a interpretação dos dados para um formato compatível, os mesmos são inseridos em suas respectivas mensurações. Da mesma forma, A API atende a requisições da ferramenta de visualização analítica, que requisita as métricas necessárias para renderização de gráficos e análise de condições para disparo de alertas.

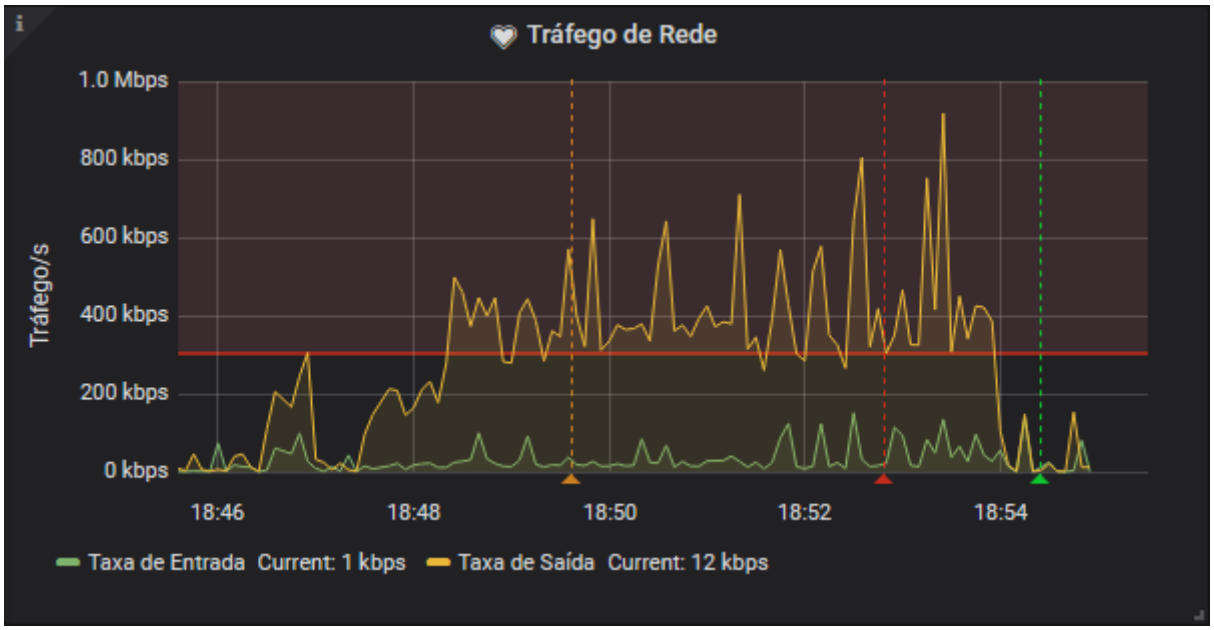
Para o disparo de alertas, o Grafana compara, em intervalos configurados, o resultado de consultas executadas com limites previamente estabelecidos. Como forma de evitar falsos-positivos, empregou-se o modo ‘repetição’.

No modo ‘repetição’, métricas podem ter três status distintos: *OK*, *Pending* e *Alert*. Se uma determinada métrica excedeu o valor máximo configurado, o status da mesma vai de *OK* para *Pending*. Se ela continua excedendo o valor por uma quantidade de vezes previamente

configurada, vai de *Pending* para *Alert*. Neste status, o alerta é disparado para o canal de comunicação configurado. Ainda, no momento em que a métrica estiver novamente dentro de valores aceitáveis, um novo alerta é disparado, avisando da mudança de status.

As mudanças de status de métricas são indicadas por elementos visualmente distintos nos gráficos renderizados. Como pode-se ver na Figura 62, uma linha horizontal contínua vermelha indica o limite entre valores aceitáveis e valores passíveis de alerta. Nesta mesma figura, também pode-se ver três linhas verticais tracejadas: a linha laranja indica o momento em que o status de uma métrica alternou de *OK* para *Pending*; a linha vermelha indica o momento em que o status foi de *Pending* para *Alert*; e a linha verde indica o momento em que o status voltou para *OK*, ficando dentro dos valores aceitáveis.

Figura 62 - Elementos indicadores de alerta



Fonte: Autoria própria

A Figura 63 exibe o alerta enviado pela ferramenta ao usuário, referente ao mesmo pico de utilização de tráfego representado pela figura anterior. As mensagens de alerta possuem 5 elementos básicos: gráfico, status, título, mensagem e URL.

Figura 63 - Exemplo de alerta enviado via Telegram

Fonte: Autoria própria

O gráfico é uma imagem que representa o exato momento em que o alerta foi disparado; o status, que aparece entre colchetes, demonstra o estado da métrica; o título é o nome do gráfico, cadastrado no dashboard em questão; a mensagem é o texto pré-cadastrado no momento da configuração do alerta; e a URL é um link que permite acessar o painel diretamente através de um navegador compatível. É importante considerar que o acesso à URL exige as credenciais de usuário do Grafana.

5.5 MODELO DE DADOS

Como previamente demonstrado, o modelo de dados criado para o armazenamento de métricas coletadas segue um padrão no que tange a nomenclatura de campos e mensurações. A Figura 64 exemplifica, através de tuplas e colunas.

Figura 64 - Exemplo de modelo das métricas de utilização de CPU

time	host	instance	type	type_instance	value
2019-08-10T23:52:02.375502273Z	acheflores	3	cpu	wait	1337767
2019-08-10T23:52:02.375503231Z	acheflores	3	cpu	interrupt	0
2019-08-10T23:52:02.375504194Z	acheflores	3	cpu	softirq	52720
2019-08-10T23:52:02.375506274Z	acheflores	3	cpu	steal	29218
2019-08-10T23:52:02.95483885Z	db.acheveiculos.com	0	cpu	user	10974616
2019-08-10T23:52:02.95488064Z	db.acheveiculos.com	0	cpu	system	1581147
2019-08-10T23:52:02.954893918Z	db.acheveiculos.com	0	cpu	wait	274763
2019-08-10T23:52:02.954905583Z	db.acheveiculos.com	0	cpu	nice	1768

Fonte: Autoria própria

Como se pode ver, a mensuração possui um campo '*time*', criado automaticamente pelo InfluxDB e que serve para o armazenamento de *timestamp*, e até 4 campos de chave (ou *Tag Keys*) os quais possuem sempre a mesma nomenclatura. No caso da mensuração de utilização de CPU, o campo '*host*' armazena o *hostname* da máquina, o campo '*instance*' armazena o *core* (núcleo) a que se relaciona o valor, o campo '*type*' possui sempre o mesmo valor (cpu), e o campo '*type_instance*' armazena o tipo de tarefa. Por último, o campo '*value*' armazena a métrica coletada, sempre no formato '*float'*.

A Figura 65 ilustra um segundo exemplo, desta vez fazendo uso da mensuração de entrada de dados via rede.

Figura 65 - Exemplo de modelo das métricas de entrada de dados

time	host	instance	type	value
2019-08-24T22:34:22.376625866Z	acheflores	eth0	if_errors	0
2019-08-24T22:34:22.959569386Z	db.acheveiculos.com	ens3	if_packets	7756462
2019-08-24T22:34:22.959575841Z	db.acheveiculos.com	ens3	if_octets	741245016
2019-08-24T22:34:22.959584526Z	db.acheveiculos.com	ens3	if_errors	0
2019-08-24T22:34:22.959587506Z	db.acheveiculos.com	ens3	if_dropped	0
2019-08-24T22:34:22.959589871Z	db.acheveiculos.com	lo	if_packets	0
2019-08-24T22:34:22.959591422Z	db.acheveiculos.com	lo	if_octets	0
2019-08-24T22:34:22.959592129Z	db.acheveiculos.com	lo	if_errors	0
2019-08-24T22:34:22.959592619Z	db.acheveiculos.com	lo	if_dropped	0
2019-08-24T22:34:22.959593795Z	db.acheveiculos.com	ens7	if_packets	142233815
2019-08-24T22:34:22.959594372Z	db.acheveiculos.com	ens7	if_octets	25067964730
2019-08-24T22:34:22.959595715Z	db.acheveiculos.com	ens7	if_errors	0
2019-08-24T22:34:22.959600725Z	db.acheveiculos.com	ens7	if_dropped	0
2019-08-24T22:34:23.927522503Z	sistema.prosystem.srv.br	eth0	if_packets	148800191
2019-08-24T22:34:23.927525836Z	sistema.prosystem.srv.br	eth0	if_octets	63099327671

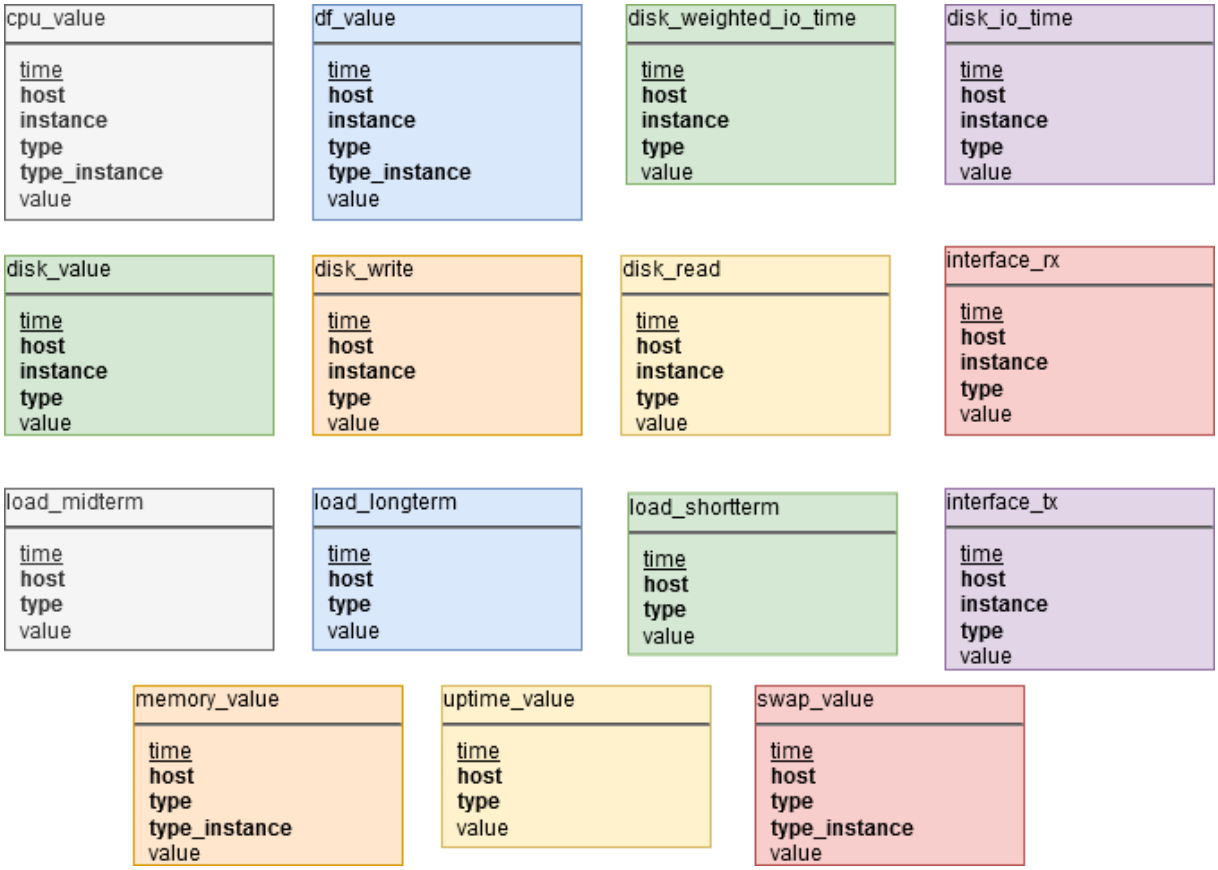
Fonte: Autoria própria

Como se pode ver, neste caso os campos '*time*', '*host*' e '*value*' tem a mesma função do exemplo anterior, o campo '*type_instance*' não é empregado, o campo '*instance*' armazena

o nome da interface de rede a que se relaciona a métrica coletada e o campo ‘*type*’ armazena a categoria em que o valor se encaixa: erros, pacotes e octetos.

A Figura 66 representa um diagrama entidade-relacionamento da implementação.

Figura 66 - Diagrama entidade-relacionamento da implementação



Fonte: Autoria própria

5.6 CONSIDERAÇÕES FINAIS

A implementação da solução proposta mostrou-se viável e funcional, sendo que não houve quaisquer impedimentos para a realização da mesma no contexto da organização AgenciaNet, de Flores da Cunha.

O fluxo de comunicação entre as ferramentas integradas foi estabelecido, e foi possível criar todos os controles previamente levantados durante a etapa de seleção de métricas, bem como foi possível implementar todos os casos de uso propostos.

6 TESTES E RESULTADOS

Após o desenvolvimento da integrações dos *softwares* selecionados (processo que envolve a preparação dos ambientes, instalação e parametrização das ferramentas, configuração de paineis de alertas e início da coleta de métricas), foram realizados testes de acordo com o que fora proposto no subcapítulo 4.5 (Validação da Solução). O escopo do processo de validação foi definido com os seguintes objetivos:

- Garantir que todos os casos de uso implementados estejam funcionando de acordo com o planejado;
- Garantir que a integração funcione de forma satisfatória, independente da quantidade de métricas coletadas (desde que dentro da política de retenção selecionada). Isto poderá também confirmar a correta utilização de alguns dos parâmetros utilizados, como o tempo de intervalo de coleta;
- Garantir que o consumo de recursos computacionais por parte da integração das ferramentas não prejudique o contexto em que foi implementada;
- Garantir que os dados coletados estejam de acordo com os dados reais das máquinas monitoradas.

O contexto da validação é o mesmo da implementação: a empresa AgênciaNet, de Flores da Cunha. Foram selecionados 5 servidores Linux que são utilizados para a execução de projetos internos da organização, cujas métricas passaram a ser coletadas após a implementação do projeto.

6.1 TESTES DE FUNCIONALIDADES DA INTEGRAÇÃO

Para a avaliação das funcionalidades da integração, foram elaborados oito casos de testes: *Login* na ferramenta Grafana, cadastro de usuário na ferramenta Grafana, adição de novas máquinas monitoradas, troca de visualização de máquina monitorada, seleção de intervalo de tempo no *dashboard* de visualização das métricas, aumento de detalhamento das métricas nos componentes do *dashboard*, configuração de alertas em um painel de métricas do *dashboard*, recebimento de alertas via grupo no mensageiro Telegram.

6.1.1 Caso de teste "*Login* na ferramenta Grafana"

O primeiro teste realizado, o qual está representado no Quadro 17, foi "*Login* na ferramenta Grafana".

Quadro 17 - Caso de teste "*Login* na ferramenta Grafana"

ID	T-001
Nome	*Login* na ferramenta Grafana
Ator(es)	Usuário, usuário administrador
Pré-Condições	O utilizador deve possuir um usuário cadastrado na ferramenta Grafana
Procedimentos	1 – O usuário abre um *browser* de internet suportado; 2 – O usuário digita o endereço público da ferramenta (*http://metricas.agencianet.net.br:3000*); 3 – O usuário preenche os dados de autenticação (admin, mji.1992); 4 – O usuário clica em no botão "*Log In*".
Pós-Condições	O usuário é redirecionado para a tela inicial da ferramenta Grafana.

Fonte: Autoria própria

Os procedimentos foram executados fazendo uso do navegador Mozilla Firefox 66. A tela de login foi carregada sem quaisquer problemas, e foram utilizados os dados de autenticação de usuário, criado no momento da instalação da ferramenta. Após a submissão do formulário de login a pós-condição foi atingida: a tela inicial da ferramenta, onde exibe-se a seleção de *dashboards* para visualização, foi carregada.

6.1.2 Caso de teste "Cadastro de usuário na ferramenta Grafana"

O segundo caso de teste visou verificar o correto funcionamento da funcionalidade da ferramenta de visualização analítica que permite cadastrar usuários, através do envio de um convite via e-mail. É durante o cadastro do usuário que se define, por exemplo, o papel do mesmo na organização previamente cadastrada na ferramenta. O usuário pode ser "*Viewer*", "*Editor*" ou "*Admin*", ou seja, ele pode possuir permissões apenas de visualização, permissões de cadastrar, editar e excluir *dashboards* existentes, ou pode ser um administrador, com permissões de gerenciar quaisquer aspectos da ferramenta. O procedimento executado pode ser visto no Quadro 18.

Quadro 18 - Caso de teste "Cadastro de usuário na ferramenta Grafana"

ID	T-002
Nome	Cadastro de usuário na ferramenta Grafana
Ator(es)	Usuário administrador
Pré-Condições	O utilizador deve possuir permissões de gerenciamento de usuários
Procedimentos	1 – O administrador acessa o menu "*Configuration* > *Users*"; 2 – O administrador clica no botão "*Invite*"; 3 – O administrador preenche os campos "E-mail", "*Name*", seleciona um "*Role*". Respectivamente: mjisoton@ucs.br, Michel Isoton, e Admin. 4 – O administrador clica novamente no botão "*Invite*".
Pós-Condições	O usuário é cadastrado, e recebe um e-mail com as instruções para definição de senha e acesso.

Fonte: Autoria própria

Os procedimentos foram executados fazendo uso de um usuário administrador, o qual possui permissões necessárias. Após o preenchimento do formulário e submissão do mesmo, o novo usuário recebeu um e-mail com os dados de autenticação e instruções.

É importante ressaltar que, para este caso de teste ser executado com sucesso, faz-se necessária a configuração de parâmetros de servidor SMTP e autenticação do mesmo no arquivo de configuração do Grafana (grafana.ini). Caso contrário, a ferramenta não pode disparar e-mails.

6.1.3 Caso de teste "Adição de novas máquinas monitoradas"

O Quadro 19 exibe o caso de teste de adição de máquinas monitoradas.

Quadro 19 - Caso de teste "Adição de novas máquinas monitoradas"

ID	T-003
Nome	Adição de novas máquinas monitoradas
Ator(es)	Usuário administrador
Pré-Condições	O utilizador deve possuir permissões de edição de *dashboards*.
Procedimentos	1 – O administrador clica em "*Create* > *Dashboard*"; 2 – O administrador clica em "*Add Panel*"; 3 – O administrador clica em "*Add Query*" e preenche as consultas a serem realizadas para a seleção e conversão de métricas; 4 – O administrador clica no menu lateral "*Visualization*" e define os parâmetros visuais do painel adicionado; 5 – O administrador repete as etapas 2-4 para cada métrica; 6 – O administrador clica em "*Save Dashboard*" e define um nome.
Pós-Condições	Um *dashboard* referente à nova máquina é criado no grupo "*General*".

Fonte: Autoria própria

Este procedimento é complexo em sua execução, porém simples se considerados apenas os procedimentos a serem seguidos. Para cada máquina a ser monitorada, deve ser criado um novo *dashboard*. Este possui diferentes paineis, cada um exibindo uma ou mais métricas relacionadas. Para este painel renderizar corretamente os dados, faz-se necessária a parametrização de alguns aspectos, como consultas InfluxQL (relacionadas à nova máquina a ser monitorada), formato (gráfico, tabela, indicador...), unidades de medidas de cada eixo, título e rótulos, além de limites para disparo de alertas.

Na execução deste teste, o resultado foi satisfatório: a pós-condição foi atingida e um novo *dashboard* foi criado sem quaisquer impedimentos.

6.1.4 Caso de teste “Troca de visualização de máquina monitorada”

Cada máquina monitorada é exibida em um *dashboard* previamente adicionado ao Grafana. Desta forma, faz-se necessária uma forma de alternar entre paineis de diferentes máquinas monitoradas. O procedimento pode ser visto no Quadro 20, que exibe o caso de teste correspondente.

Quadro 20 - Caso de teste "Troca de visualização de máquina monitorada”

ID	T-004
Nome	Troca de visualização de máquina monitorada
Ator(es)	Usuário
Pré-Condições	O utilizador deve possuir permissões de visualização de *dashboard*.
Procedimentos	1 – O usuário clica no nome do *dashboard* que está sendo visualizado (Ex: *catalogos.agencianet.net.br*); 2 – O usuário expande o grupo de *dashboards* no qual os *dashboards* foram cadastrados previamente; 3 – O usuário clica no *dashboard* desejado (Ex: *acheveiculos.com*).
Pós-Condições	A visualização é alternada para os paineis do *dashboard* selecionado.

Fonte: Autoria própria

Na execução deste caso de teste, a pós-condição foi atingida. Há de se considerar que, para alternar entre máquinas monitoradas, faz-se necessário que haja no mínimo duas máquinas cadastradas na ferramenta. Ainda, a tela de seleção é exibida não apenas ao clicar no nome do *dashboard* atual, como também ao visualizar a tela inicial da ferramenta.

6.1.5 Caso de teste “Seleção de intervalo de tempo”

Um dos casos de uso propostos refere-se à possibilidade de alternar as métricas renderizadas por meio da seleção de faixas de tempo (sendo o intervalo entre uma data e hora de início e uma data e hora de fim). O Quadro 21 demonstra o caso de teste correspondente.

Quadro 21 - Caso de teste "Seleção de intervalo de tempo"

ID	T-005
Nome	Seleção de intervalo de tempo
Ator(es)	Usuário
Pré-Condições	O usuário deve possuir permissões de visualização de *dashboard*; O usuário deve estar visualizando um *dashboard*; A aplicação não deve estar em modo *Cycle* ou *Kiosk*;
Procedimentos	1 – O usuário clica no seletor de tempo, no canto superior direito; 2 – O usuário clica em uma faixa de tempo predefinida; 3 – Se é de interesse do usuário, na seção ‘*Custom Range’* ele pode definir uma faixa de tempo customizada; 4 – O usuário clica em ‘*Apply’*;
Pós-Condições	Os paineis são renderizados considerando a faixa de tempo selecionada.

Fonte: Autoria própria

Para este caso de uso, foi utilizado o *dashboard* referente à máquina denominada “acheflores”. Ao renderizar o *dashboard*, os procedimentos descritos foram realizados, e selecionou-se a faixa de datas de 15 de agosto de 2019 às 15:00:00 horas, até 18 de agosto de 2019 às 15:00:00 horas. Os paineis foram recarregados e as métricas foram renderizadas com sucesso, bem como exigido pelo caso de teste.

6.1.6 Caso de teste “Aumento do detalhamento de métricas”

Este caso de teste é representado pelo Quadro 22.

Quadro 22 - Caso de teste "Aumento do detalhamento de métricas"

ID	T-006
Nome	Aumento do detalhamento de métricas
Ator(es)	Usuário
Pré-Condições	O usuário deve estar visualizando um *dashboard*.
Procedimentos	1 – O usuário posiciona o cursor sobre um painel e clica na posição inicial que deseja detalhar; 2 – O usuário arrasta o cursor até o momento desejado e solta o clique;
Pós-Condições	Os paineis serão renderizados novamente com a nova faixa de tempo.

Fonte: Autoria própria

O resultado final é semelhante ao caso de teste de seleção de intervalo de tempo: é definida uma faixa de tempo, a qual é imediatamente aplicada e os paineis são renderizados novamente. Utilizou-se o *dashboard* referente à máquina “db.acheveiculos.com”, e efetuando o movimento de cursor para detalhar a faixa de tempo aproximada de 18 de agosto de 2019 às 18:00:00 horas até 18 de agosto de 2019 às 19:00:00 horas, a pós-condição foi atingida.

6.1.7 Caso de teste “Configuração de alertas em paineis”

Um painel de um *dashboard* pode possuir um ou mais alertas configurados, os quais são disparados para os canais de comunicação previamente configurados. O Quadro 23 demonstra o caso de teste correspondente à configuração destes alertas.

Quadro 23 - Caso de teste "Configuração de alertas em paineis"

ID	T-007
Nome	Configuração de alertas em paineis
Ator(es)	Usuário administrador
Pré-Condições	O usuário deve possuir permissões de edição de *dashboards*; O usuário deve estar visualizando um dos *dashboards* existentes.
Procedimentos	1 – O usuário escolhe um painel existente (Ex: Utilização de memória RAM); 2 – O usuário clica no título do painel; 3 – O usuário clica em ‘*Edit’*; 4 – O usuário clica na guia ‘*Alert’*; 5 - O usuário preenche um nome e escolhe um intervalo para avaliação da regra; 6 - O usuário seleciona condições para as consultas do painel, bem como ações a serem tomadas em caso de falhas; 7 - O usuário digita a mensagem de notificação a ser disparada em alertas; 8 – O usuário clica em ‘*Save Dashboard’* para salvar as alterações.
Pós-Condições	Um novo alerta será configurado para o canal de comunicação padrão.

Fonte: Autoria própria

Utilizando-se de um usuário com permissões de edição, foi feito o procedimento descrito, editando o painel de ‘Utilização de Memória RAM’ do *dashboard* ‘acheveiculos.com’. Visto que o canal de comunicação ‘Telegram’ fora configurado nas etapas iniciais de desenvolvimento do projeto, foi possível configurar o alerta. A regra empregada foi um cálculo de média de memória livre, avaliada em intervalos de 1 minuto durante 5 minutos. Se a mesma permanecer abaixo de 32MB, o alerta é disparado.

6.1.8 Caso de teste “Recebimento de alertas via Telegram”

O caso de teste ‘recebimento de alertas via Telegram’ complementa o caso de teste imediatamente anterior. O Quadro 24 ilustra os procedimentos necessários.

Quadro 24 - Caso de teste "Recebimento de alertas via Telegram"

ID	T-008
Nome	Recebimento de alertas via Telegram
Ator(es)	Usuário
Pré-Condições	O usuário deve possuir uma conta no serviço de mensagens instantâneas Telegram; O usuário deve fazer parte do grupo de mensagens criado previamente no Telegram; O painel de ‘Utilização de Memória RAM’ deve possuir alertas configurados com condições válidas;
Procedimentos	1 – O administrador da máquina monitorada correspondente ao *dashboard* faz um acesso remoto á máquina; 2 – O administrador executa qualquer procedimento que faça uso intenso de memória RAM; 3 – O administrador mantém a execução de forma a fazer com que a utilização de memória ultrapasse os níveis previamente configurados nos alertas do painel; 4 – Aguarda-se o tempo previamente configurado no alerta, de forma que a expressão seja avaliada com um resultado positivo, disparando o alerta.
Pós-Condições	Um alerta é disparado para o grupo de mensagens através do canal de comunicação padrão configurado (Telegram).

Fonte: Autoria própria

Para a execução deste caso de teste, foi empregada a regra de alerta implementada no caso de teste anterior (avaliação de utilização de memória RAM na máquina “acheveiculos.com” com intervalos de 1 minuto, durante 5 minutos). Para aumentar a utilização de memória RAM na máquina alvo de forma artificial, foi feito um acesso remoto e executou-se um *script* de redimensionamento de imagens, sendo esse um procedimento comumente executado na máquina em questão.

A execução do *script* em questão levou aproximadamente 7 minutos, sendo que a utilização de memória RAM caiu para um valor inferior ao mínimo aceitável. A partir do momento em que a regra avaliada para o disparo de alertas tornou-se verdadeira, o grupo de mensagens registrou o recebimento de um alerta para o painel e *dashboard* utilizados. Logo, a pós-condição de sucesso do teste foi atingida.

6.2 ANÁLISE DE UTILIZAÇÃO DE RECURSOS DA INTEGRAÇÃO

Como definido no capítulo 4.5, para validar a proposta foi feita a análise de métricas referentes à utilização de recursos computacionais por parte das ferramentas integradas. Este tipo de análise é importante devido ao fato de que a solução proposta foi instalada em um ambiente cujos recursos são limitados, característica comum em pequenas organizações. Ainda, considerando que este tipo de ferramental tende a exigir uma maior capacidade computacional, faz-se necessário considerar que a viabilidade da solução pode ser comprometida à medida que a quantidade de métricas coletadas e a utilização da solução aumenta.

Para atestar que a solução não teve um impacto negativo no ambiente, inicialmente foi analisado o tráfego de entrada de dados da máquina gerente. Isto permitiu não apenas conhecer a quantidade de dados para este estudo de caso, como também foi possível calcular o impacto que um possível aumento na quantidade de servidores monitorados causará.

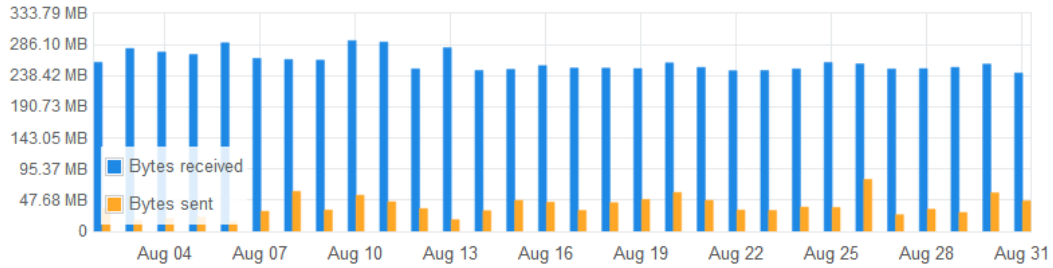
Além disso, calculou-se a ocupação de espaço em disco para a solução implementada, e fazendo uso de cálculos, foi analisada a possível total utilização de métricas, levando em consideração a política de retenção implementada.

Por fim, como forma de atestar a usabilidade da ferramenta de visualização analítica, foram coletados os tempos de carregamento dos paineis de *dashboards*, os quais devem ser rápidos o suficiente para não afetarem o fluxo de trabalho do usuário da solução.

6.2.1 Tráfego enviado ao ambiente gerente

Pôde-se coletar as estatísticas de tráfego de entrada e saída de dados do ambiente gerente analisando o adaptador de rede virtual, o qual fornece à máquina virtualizada a capacidade de conectividade via NAT. Sumarizou-se tais estatísticas de tráfego na Figura 67.

Figura 67 - Tráfego de rede na máquina gerente durante 30 dias



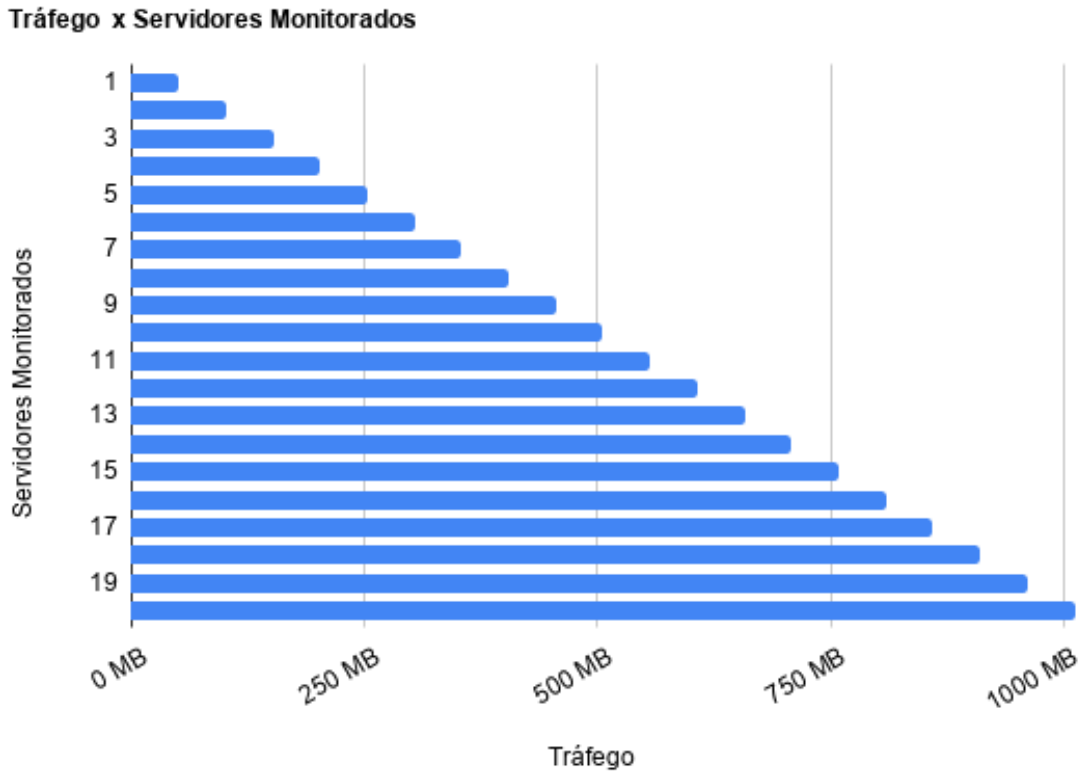
Fonte: Autoria própria

Visto que a solução foi implementada em uma máquina virtualizada no ambiente gerente, e tal máquina tem como único objetivo a hospedagem da solução proposta neste trabalho, pode-se considerar que tais métricas dizem respeito única e exclusivamente à solução proposta neste trabalho.

Como pode-se ver no gráfico, a média de tráfego diário na máquina gerente é de 252,9 MB. Considerando o mês de agosto de 2019, o tráfego total foi de 7,84 GB. Isto significa que o monitoramento de um único servidor resulta em 50,58 MB de tráfego recebido diariamente.

Como pode ser visto na Figura 68, pode-se estabelecer uma razão previsível entre a quantidade de servidores monitorados e o tráfego de entrada de dados esperado. Isto se deve ao fato de que todos os servidores coletam exatamente as mesmas métricas e empregam o mesmo *software* coletor.

Figura 68 - Razão entre servidores monitorados e tráfego de rede



Fonte: Autoria própria

No que tange a taxa de saída de dados via rede, tal métrica não é considerada relevante para análise visto que tanto o banco de dados de séries temporais (InfluxDB) como a ferramenta de visualização analítica (Grafana) se encontram no mesmo ambiente, o que faz com que a comunicação entre os mesmos ocorre localmente. Todo o tráfego de saída mensurado (uma

média de 48 MB diários) refere-se ao envio de dados pelo servidor Apache, criado pelo Grafana para a utilização dos paineis via interface WEB.

Ao analisar as métricas coletadas, pode-se concluir que a solução proposta não tem um impacto negativo no ambiente em que está inserida. Levando em consideração o monitoramento de cinco servidores, o tráfego diário é quase que imperceptível, considerando as outras aplicações em execução no mesmo ambiente. Ainda, é possível aumentar o número de servidores monitorados sem ter um grande impacto na métrica.

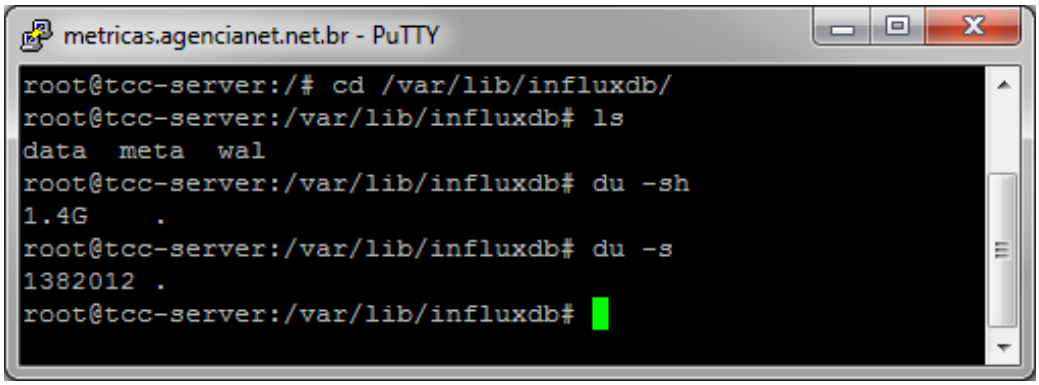
6.2.2 Espaço ocupado pelas métricas

Para a coleta deste dado, pode-se fazer uma relação entre o tamanho total do banco de dados e o tempo corrido desde a implementação.

O diretório de armazenamento de dados do InfluxDB possui três outros diretórios: *meta*, *wal* e *data*. O primeiro possui informações referentes á metadados dos bancos de dados criados; o segundo é responsável por armazenar dados recém recebidos e não comprimidos, que quando atingem um certo tamanho (definido pelo parâmetro *cache-snapshot-memory-size*) são comprimidos e movidos; e o terceiro armazena os dados de forma definitiva após a compressão, ficando armazenados enquanto não forem removidos manualmente ou automaticamente por uma política de retenção, se definida.

Como pode-se ver na Figura 69, após 20 dias do início da coleta de métricas o banco de dados está consumindo 1382012 *kilobytes*.

Figura 69 - Utilização de espaço das métricas



Fonte: Autoria própria

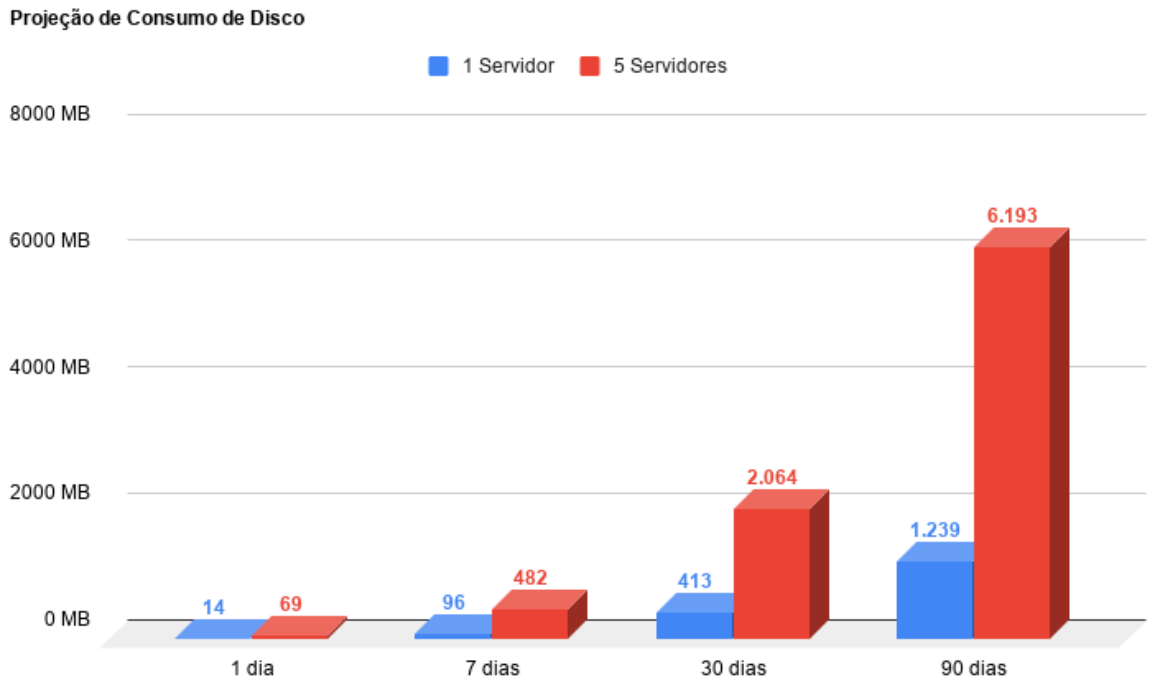
No momento em que o comando *du –s* foi empregado, a implementação completou exatas 482 horas. Considerando que estão sendo monitorados 5 servidores e que a granularidade

das coletas é de 5 segundos, pode-se inferir que as métricas de cada máquina monitorada ocupam um total de 573,44 KB por hora.

Isto significa que no momento em que a política de retenção do banco de dados entrar em vigor (ou seja, quando o equivalente a 90 dias de métricas estiverem armazenadas), o banco de dados de séries temporais estará consumindo 6.193 MB.

A Figura 70 demonstra uma projeção de utilização de disco, calculada com base no consumo por hora.

Figura 70 - Projeção de consumo de disco



Fonte: Autoria própria

Como fora especificado no capítulo 4.1, a solução foi implementada em uma máquina virtual que dispõe de 15 GB de armazenamento. Visto que o sistema operacional e dependências consomem cerca de 3,5 GB, pode-se concluir que desde que as políticas de retenção estejam funcionais, a máquina gerente jamais irá exceder o limite de uso de disco.

Em 90 dias, o monitoramento de 5 servidores irá ter consumido 6.193 MB (ou 6,19 GB), de forma que a utilização máxima de disco no ambiente preparado ficará estagnada em aproximadamente 9,7 GB, o que faz com que a implementação da solução no ambiente deste estudo de caso viável.

6.2.3 *Delay* na renderização de gráficos

O processo de renderização de paineis no Grafana (sejam gráficos de linhas, pontos, texto ou indicadores) é constituído pela mesma sequência: seleção de faixa de datas, envio de parâmetro ao *back-end* da aplicação, execução de *query* no InfluxDB, retorno de dados em JSON, e renderização do painel, a qual faz uso de bibliotecas javascript.

Para cada painel presente no *dashboard* visualizado, executa-se uma requisição. Isto significa que o *back-end* precisa responder 11 requisições simultaneamente, para cada vez que o *dashboard* for atualizado.

Fazendo uso das ferramentas de desenvolvedor do navegador Mozilla Firefox (Figura 71), pode-se contabilizar o tempo que cada requisição leva para ser respondida. Tal tempo corresponde à diferença entre o exato momento em que a requisição foi enviada ao servidor e o exato momento em que a resposta foi recebida pelo navegador, desprezando o tempo gasto em outras tarefas.

Figura 71 - Ferramentas de desenvolvedor (Rede)

Fonte: Autoria própria

Visto que o tamanho da amostra de dados retornados tem influência no tempo de retorno das requisições (seja pelo tempo gasto no processamento da consulta e/ou pelo transporte dos dados via rede), foram coletadas métricas em 4 situações distintas: visualização de métricas com um intervalo de 1 hora, 1 dia, 1 semana e 1 mês. O Quadro 25 exibe os resultados obtidos.

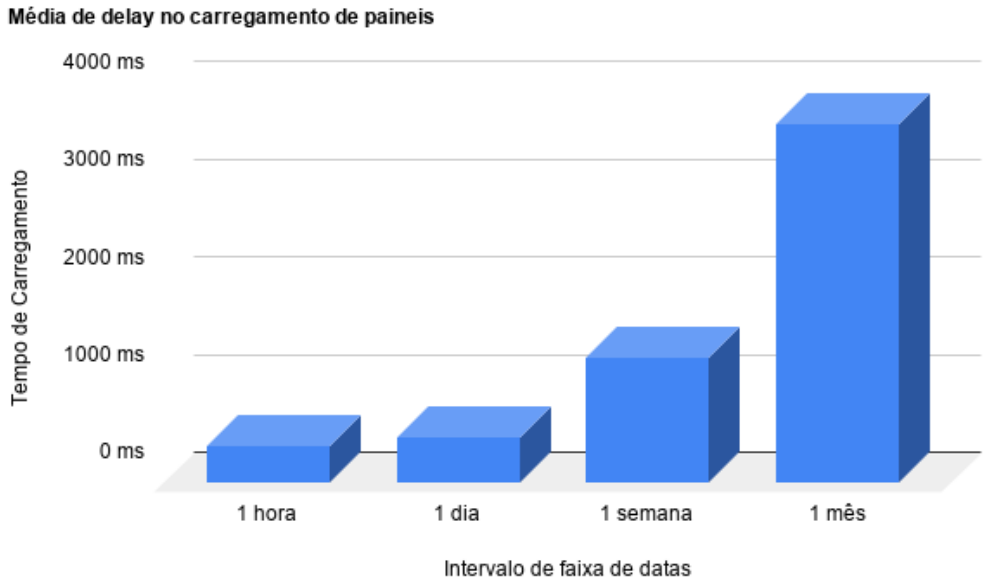
Quadro 25 - *Delays* de renderização de paineis

Painel	**1 hora**	**1 dia**	**1 semana**	**1 mês**
Uptime	154 ms	166 ms	229 ms	795 ms
Tráfego de Rede	188 ms	313 ms	451 ms	1759 ms
Utilização de CPU	398 ms	656 ms	2299 ms	5503 ms
Utilização de Memória RAM	449 ms	519 ms	1373 ms	4900 ms
Utilização de Memória SWAP	246 ms	298 ms	644 ms	1488 ms
Tráfego de Disco	380 ms	419 ms	1262 ms	3148 ms
Operações em Disco	299 ms	422 ms	1284 ms	4379 ms
Utilização de Disco	528 ms	527 ms	1870 ms	4458 ms
CPU 1m	521 ms	453 ms	1185 ms	4357 ms
CPU 5m	524 ms	658 ms	1803 ms	4801 ms
CPU 15m	514 ms	664 ms	1807 ms	4921 ms

Fonte: Autoria própria

Como pode-se ver, o tempo de carregamento dos paineis varia conforme o tamanho do intervalo de datas a serem renderizados: quanto maior o intervalo, mais tempo o *back-end* da aplicação demora para processar e responder a requisição. Ainda, alguns paineis são naturalmente mais rápidos devido às funções que utilizam e à quantidade métricas necessárias para a renderização. A Figura 72 sumariza os tempos médios de carregamento.

Figura 72 - Média de *delay* no carregamento de paineis



Fonte: Autoria própria

Como ilustrado no gráfico da Figura 72, ao buscar por um intervalo de 1 hora de métricas a média de tempo de carregamento dos paineis é de 382 ms. Ao analisar um intervalo de 24 horas de métricas, este tempo é de 483 ms, e um intervalo de uma semana leva 1242 ms para renderizar. Estes são os intervalos naturalmente mais empregados para a visualização de métricas, e os *delays* são aceitáveis: não chegam a prejudicar a utilização da ferramenta e permitem a alternância rápida entre intervalos de tempo e *dashboards*.

Ao buscar um intervalo de 1 mês de métricas, a média de tempo de carregamento é de 3683 ms. Embora alto, refere-se a um intervalo de tempo raramente empregado, e pode ser desconsiderado.

6.3 CONSIDERAÇÕES FINAIS

Os testes de validação foram executados conforme o que fora previamente planejado. Verificou-se que todos os casos de testes tiveram suas pós-condições alcançadas, e todas as métricas de utilização de recursos tiveram resultados satisfatórios, considerando a execução em um banco de dados populado com 30 dias de métricas reais.

Pode-se constatar ainda que todos os requisitos funcionais planejados foram atendidos, e que o parâmetro de intervalo da coleta de métricas (5 segundos) é compatível com o objetivo da solução e com as limitações impostas pelo contexto de implantação.

7 AVALIAÇÃO DA SOLUÇÃO

A solução proposta fora planejada levando em consideração que sua utilização servirá como mais um recurso de suporte ao profissional administrador de sistemas. Não se trata de uma ferramenta cujo objetivo é resolver por definitivo a ocorrência de incidentes desta natureza; visa garantir ao profissional a capacidade de tomar conhecimento de situações-problema antes mesmo da ocorrência de incidentes, e agir de acordo.

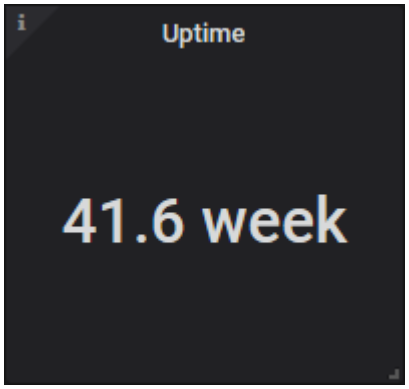
Durante o levantamento de métricas a serem empregadas, levou-se em consideração a relevância das mesmas na análise de situações, no *debugging* de aplicações e no correto funcionamento do ambiente. Além disso, fez-se necessário considerar a melhor forma de apresentar tais dados de forma a facilitar a interpretação gerando informações válidas que possibilitem uma reação rápida.

7.1 INDICADOR DE *UPTIME*

O indicador de *uptime*, o qual apresenta em um formato textual o tempo desde o último *boot*, tem o objetivo de informar sobre interrupções acidentais ou propositais no funcionamento do ambiente, o qual pode ser responsável por um ou mais serviços onde a disponibilidade tem extrema importância.

Embora não se possa estabelecer uma relação direta entre o tempo de *uptime* e a estabilidade do serviço, ambientes que estão em execução há mais tempo mantém serviços disponíveis por mais tempo. A Figura 73 exibe um indicador de *uptime* de um servidor com mais de 41 semanas de uptime.

Figura 73 - Indicador de *uptime* de mais de 41 semanas



Fonte: Autoria própria

O *uptime* ainda pode ser um bom indicador de que pode ser necessário o *reboot* do ambiente. Frequentemente, com o objetivo de manter o ambiente o mais estável possível, atualizações importantes de dependências e do *kernel* do sistema são ignoradas; se não ignoradas, são instaladas em tempo de execução e sem o reinício do sistema. Isto evita que atualizações importantes sejam aplicadas.

Considerando que aproximadamente a cada 2 ou 3 meses, uma nova base do *kernel* é lançada (KERNEL, 2019), este pode ser um bom momento para um *reboot* programado do sistema.

7.2 INDICADORES DE TAMANHO DE FILA DE EXECUÇÃO

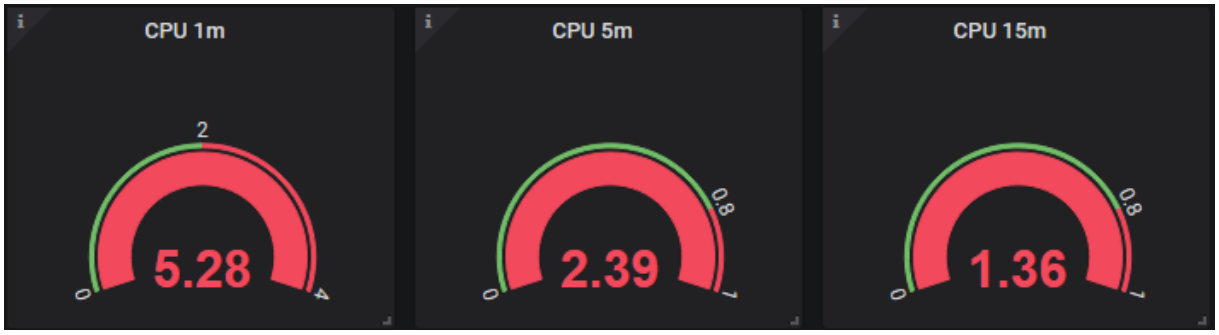
Os indicadores de tamanho de fila de execução (*run-queue length*) são frequentemente empregados para detectar gargalos no processamento em servidores de natureza multitarefa, ou de servidores com múltiplas aplicações em execução simultaneamente. Este tipo de métrica é comumente analisada levando em consideração três amostras distintas: 1, 5 e 15 minutos. Em situações normais, atribui-se maior importância ao indicador de 15 minutos devido à sua maior amostra, que lhe permite refletir com mais veracidade a situação atual do servidor.

A fila de execução possui processos ativos. Cada processo possui prioridades diferentes, as quais são empregadas pelo escalonador para determinar qual processo será executado. Além disso, processos podem voltar à fila se não finalizados, e podem ser removidos da mesma se estão em espera, interrupção ou foram finalizados.

Durante a validação da solução, um problema recorrente em uma das aplicações críticas da organização foi identificado graças a este indicador. Frequentemente, durante o horário de pico (20:00 e 22:00 horas), o tempo de resposta das requisições a uma determinada API esteve acima de 2500 ms constantemente. Ao analisar a fila de execução para estes horários, pode-se detectar um número alto de tarefas: mesmo considerando a maior amostra, a quantidade excedeu o limite pré-estabelecido, como pode ser visto na Figura 74.

Tratando-se de uma máquina *single-core*, o servidor em questão pode executar apenas uma tarefa (processo) simultaneamente. Isto significa que requisições à API não foram executadas imediatamente ao serem recebidas: precisaram ser escalonadas de acordo com a prioridade atribuída.

Figura 74 - Indicadores de fila de execução com muitos processos

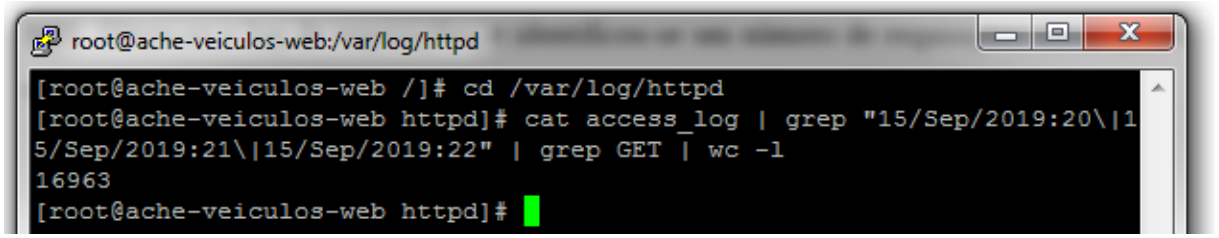


Fonte: Autoria própria

Como pode ser visto na Figura 74, considerando apenas o curto prazo a média de tarefas na fila de execução manteve-se bem acima da capacidade da máquina. No médio e longo prazo a média de tarefas diminuiu, porém sempre esteve acima de 1, o que indica que na média, sempre houve tarefas em espera na fila de execução durante o período da amostra coletada.

Fazendo uso da informação provida pela solução (ilustrada na Figura 74) e conhecendo a natureza das tarefas executadas pelo servidor, o profissional responsável pela aplicação confirmou a suspeita de sobreuso da aplicação através da coleta de evidência. Acessando via SSH, filtrou o *log* de requisições, e identificou um número de requisições maior que o suportado pela infraestrutura alocada. Isto pode ser visto na Figura 75.

Figura 75 - Evidência de número de requisições alto



Fonte: Autoria própria

Neste caso, foi feito o provisionamento de mais núcleos de processamento para garantir que as requisições sejam processadas e retornadas em um tempo inferior a 500 ms, tempo acordado pela organização aos clientes.

Logo, pode-se concluir que estes indicadores garantem à organização a possibilidade de identificar subusos e sobreusos de recursos de processamento em determinadas faixas de tempo, o que pode indicar tanto problemas em tempo de execução, como erros durante o provisionamento inicial de recursos computacionais. Porém, deve-se ter a percepção de que os mesmos ilustram uma amostra limitada do comportamento do servidor: é comum e

completamente aceitável que o tamanho da fila de execução seja maior em momentos de maior carga, como durante a execução de tarefas agendadas ou em horários de pico.

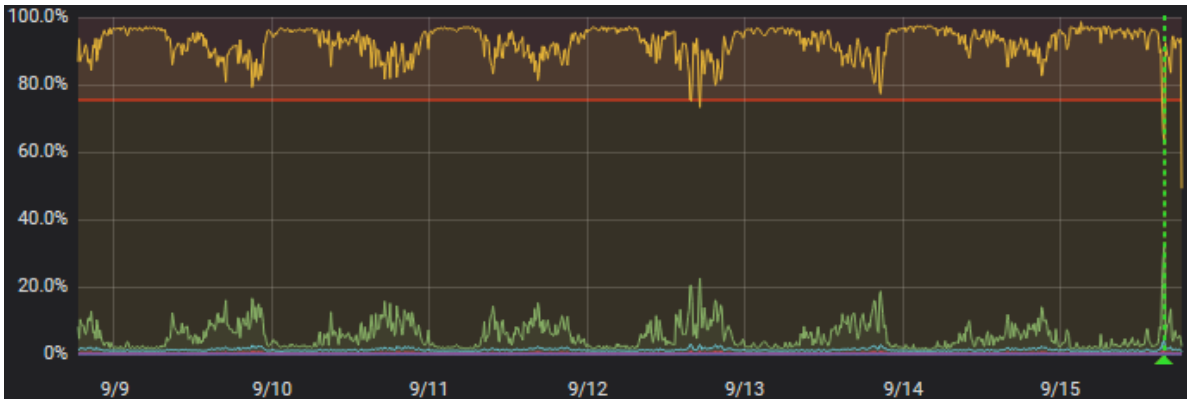
7.3 GRÁFICO DE USO DE PROCESSAMENTO

O gráfico de uso de processamento exibe o tempo gasto pelo processador para executar diferentes tipos de tarefas. Desta forma, através da análise deste gráfico, pode-se conhecer diferentes aspectos da execução de tarefas e do consumo de recursos do servidor.

Além da identificação de momentos de pico de processamento, pode-se conhecer a natureza das tarefas que mais consomem recursos. Em ambientes virtualizados, pode-se identificar a utilização de processamento por parte do *hypervisor* (*steal*), e também é possível identificar gargalos na leitura e escrita de dados, através do tempo gasto em tarefas deste tipo (*wait*). É possível ainda identificar interrupções causadas por dispositivos de *hardware*.

Durante a utilização da ferramenta, foi possível concluir que a utilização de uma grande amostra para análise de uso de processamento neste gráfico tem um efeito reverso se comparado ao indicador analisado anteriormente: quanto maior a amostra, mais difícil é identificar possíveis incidentes. A Figura 76 exibe uma amostra de uma semana de métricas renderizadas.

Figura 76 - Tipos de tarefas consumindo tempo de processamento



Fonte: Autoria própria

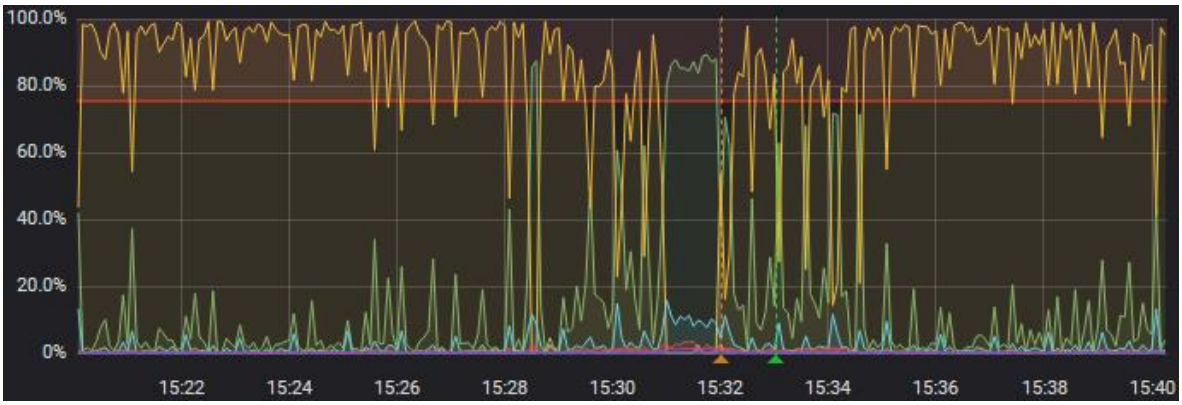
Como é possível visualizar, momentos de sobreuso e subuso são normalizados de acordo com o agrupamento de métricas executado nas consultas InfluxQL, o qual varia de acordo com o tamanho da amostra solicitada pelo usuário. Isto significa que possíveis picos de utilização não aparecem no gráfico ao empregar grandes amostras.

A utilização durante o período de validação da solução ainda revelou que é comum que o processamento de servidores suba para 100% em curtos momentos. Isto ocorre normalmente

durante o *boot* do servidor, ou em momentos em que serviços e processos são inicializados, e logo a utilização se normaliza.

Para fins de recebimento de alertas, foi configurada a necessidade de utilização de 75% do tempo de processamento por tarefas *user* e *system*, durante 2 minutos consecutivos. Considerando que passados 60 dias desde o início da coleta de amostras, nenhum alerta desta métrica foi enviado ao usuário da solução, o gráfico permitiu à organização concluir que momentos de pico de processamento tendem a durar pouco tempo, não sendo necessária intervenção. A figura 77 ilustra um destes picos de curta duração, ao mesmo tempo que demonstra como uma amostra de tempo menor (neste caso, 1 hora) permite uma identificação mais clara dos níveis de utilização de processamento.

Figura 77 - Amostra de 20 minutos de processamento



Fonte: Autoria própria

Como é possível ver na figura acima, durante um momento de pico de utilização de uma tarefa de *user* (linha verde), a ferramenta chegou a identificar uma utilização de recursos alta (linha vertical laranja), porém tal utilização não se manteve por tempo o suficiente para o disparo de alerta. Logo após, o uso de processamento foi normalizado (linha vertical verde).

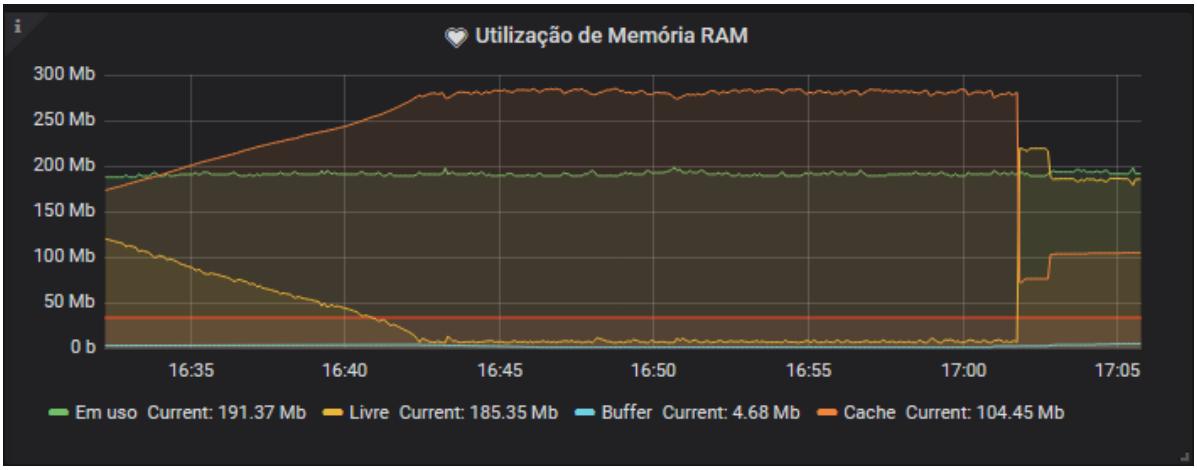
7.4 GRÁFICO DE UTILIZAÇÃO DE MEMÓRIA RAM

Como o nome sugere, este gráfico exibe as métricas de utilização de memória RAM. Através das variações destas métricas, pode-se identificar momentos de instabilidade dos serviços fornecidos por um servidor, onde a alocação de memória pode não suprir as necessidades das aplicações em execução.

Durante a etapa de validação, foi possível identificar ocasiões em que o gráfico representou métricas inesperadas. Embora frequentes, as variações de utilização de memória

RAM tendem a ser graduais de acordo com a execução e encerramento de aplicações que alocam tal recurso; porém, houve variações bruscas. A Figura 78 demonstra um destes momentos, em que pode-se perceber linhas verticais, que indicam uma queda ou aumento instantâneo de utilização de memória.

Figura 78 - Situação de falta de memória RAM



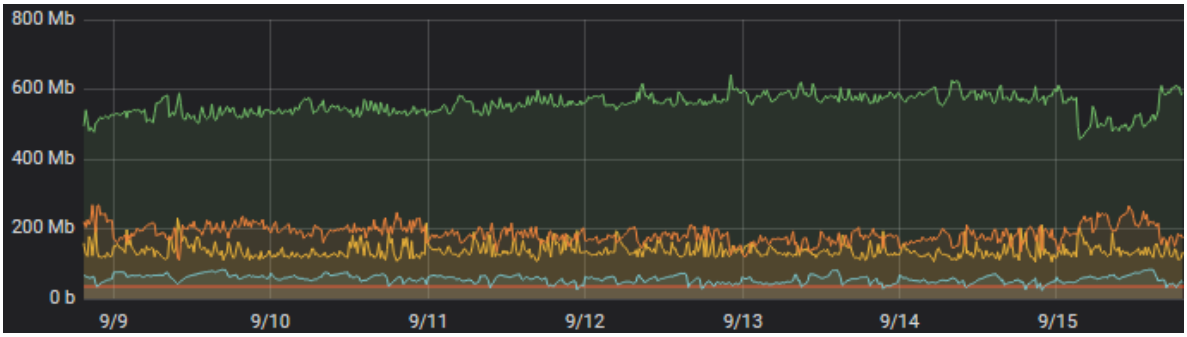
Fonte: Autoria própria

A figura acima ilustra uma situação de alocação crescente de memória, a qual culminou no esgotamento da mesma. Durante aproximadamente 20 minutos, a utilização de memória se estabilizou, e no momento 17:02 o OOM Killer entrou em ação, encerrando o processo alocador. A partir daí, a utilização de memória RAM voltou aos seus valores habituais.

Na análise deste gráfico, é fundamental considerar que, no contexto de sistemas operacionais UNIX, memória disponível é a soma de memória livre, *buffer* e *cache*. Ainda, visto que a métrica *cache* é o espaço alocado para o *cache* de operações em disco (*disk caching*), esta é mais uma informação visível no gráfico, a qual pode indicar a causa de uma possível lentidão no acesso a arquivos armazenados em disco.

Pode-se ainda concluir que os efeitos da normalização devido ao agrupamento de métricas também afeta este gráfico, porém não tão agressivamente, como pode ser visto na Figura 79.

Figura 79 - Intervalo de uma semana de métricas de memória RAM



Fonte: Autoria própria

Durante o período de coleta de métricas, uma das máquinas foi configurada de forma que se a média de memória livre ficasse abaixo de 64 MB, um alerta seria enviado através do canal de comunicação configurado. Isto permitiu que a organização constatasse um fato antes desconhecido: a utilização excessiva de memória RAM causava interrupções no fornecimento de um dos serviços, as quais não eram percebidas e não eram reportadas. A Figura 80 exibe um dos alertas disparados.

Figura 80 - Alerta enviado durante o período de validação da solução

Fonte: Autoria própria

Tratando-se de um servidor executando uma aplicação WEB, foram ajustados os valores de utilização máxima de memória por requisição, o que resolveu o problema.

7.5 GRÁFICO DE UTILIZAÇÃO DE MEMÓRIA SWAP

O gráfico de utilização de memória SWAP exibe a capacidade total de memória, o total de memória ocupada e o total disponível. Visto ser empregada apenas em situações onde a memória RAM tenha atingido níveis críticos ou possa ter se esgotado, sua utilização tende a indicar a falta de recursos computacionais suficientes para suprir a demanda de alocação de memória das aplicações.

Devido à natureza do processo de *swapping*, a alta utilização de memória SWAP pode ser a causa de instabilidades na leitura e escrita de disco: a constante leitura e escrita de dados entre disco e memória RAM toma muito tempo dos dispositivos, os quais demoram para responder e desempenhar suas funções convencionais.

Durante o período de validação da solução, nenhum dos 5 servidores monitorados fez uso de memória SWAP, motivo pelo qual não foi possível analisar este tipo de métrica. A decisão da não utilização deste tipo de memória é decorrente de situações internas da organização, as quais não são relevantes para a análise da solução.

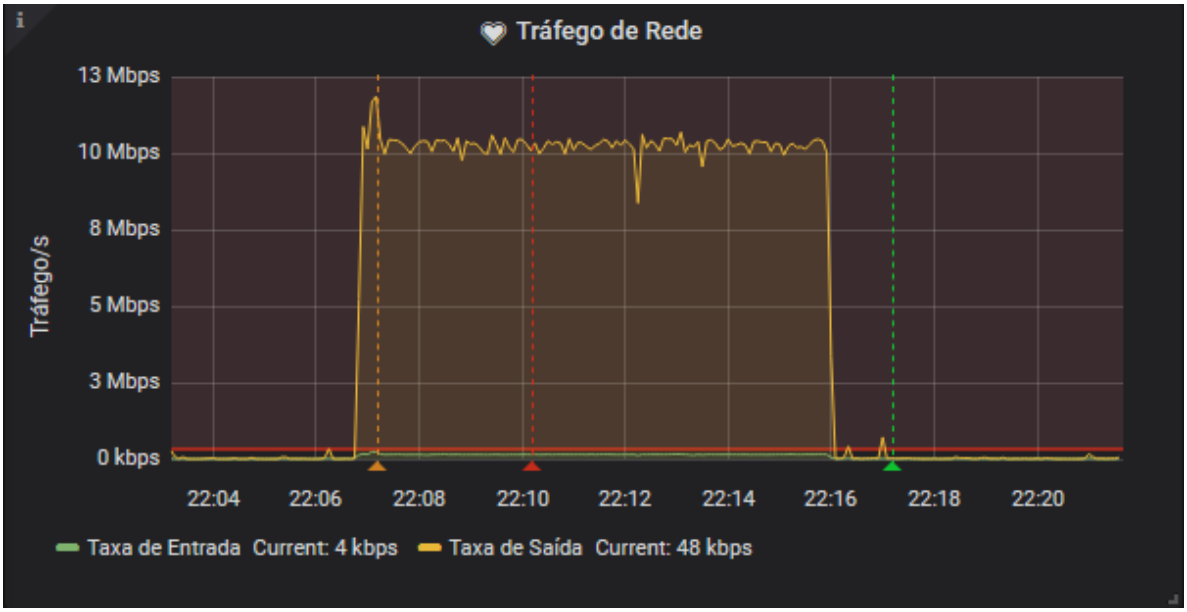
7.6 GRÁFICO DE TRÁFEGO DE REDE

Este gráfico ilustra a taxa de entrada e saída de dados via interface de rede. Os valores registrados podem se referir tanto ao tráfego para a rede interna como para a rede externa.

A análise deste gráfico permite a identificação de situações atípicas. Por exemplo, o tráfego excessivo e não-convencional de entrada de dados pode indicar a utilização indevida do servidor por um invasor, o qual se faz valer de brechas de segurança para baixar *malwares*. O tráfego excessivo e não-convencional de saída de dados pode indicar a exploração de brechas de segurança, ou algo legítimo e comum como a execução de rotinas de *backup*.

Na Figura 81, é possível ver que o tráfego deste servidor, normalmente ínfimo, sobe até 10 Mbps e mantém-se neste patamar durante 9 minutos. Este é o tempo necessário para que os arquivos de *backup* sejam transferidos para um ambiente externo. Durante este processo, não apenas o tráfego de rede, como também a utilização de processamento, memória RAM, tráfego de disco e número de operações em disco são afetados.

Figura 81 - Tráfego de rede durante rotina de *backup*



Fonte: Autoria própria

No contexto da organização onde a solução fora implantada para validação, o controle sobre o tráfego de dados é muito importante devido às limitações do *link* de conexão. Devido a isso, foi estabelecido que para o servidor da figura 81, uma situação em que as taxas de entrada e/ou saída acima de 300 Kbps durante mais de 3 minutos ininterruptos causaria o envio de alerta. A Figura 82 ilustra um dos alertas enviados.

Figura 82 - Alerta referente ao tráfego de rede não convencional

Fonte: Autoria própria

Na figura 82, pode-se perceber um período de tempo onde a taxa de saída de dados excedeu os limites estabelecidos, o que ocasionou o disparo do alerta. O tráfego de entrada de dados também atingiu valores percebidamente altos, porém não o suficiente para o disparo de alertas.

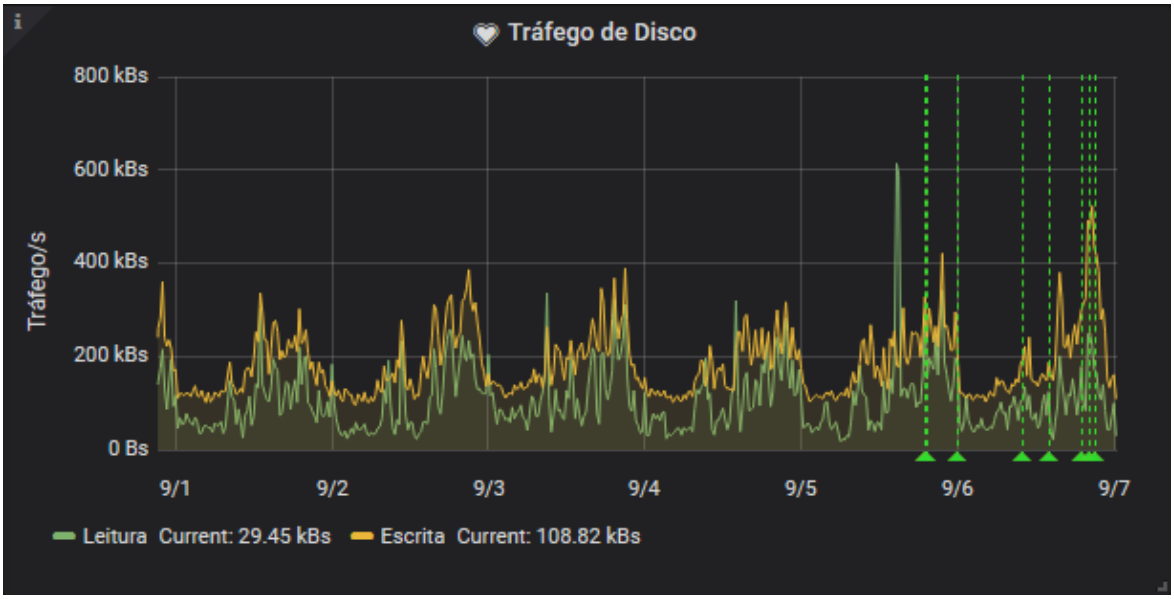
Uma conclusão na qual pode-se chegar foi que, ao analisar o gráfico de tráfego de rede, deve-se considerar a velocidade da interface de rede e do *link* de *internet* como fatores limitantes: taxas muito próximas do limite indicam um possível gargalo de recursos, o que tende a causar problemas no fornecimento de serviços. Ainda, embora óbvio, pode-se citar que uma alta taxa de entrada e saída de dados normalmente pode ser refletida também em variações nas taxas de leitura e/ou escrita de disco, embora isto não seja obrigatório.

7.7 GRÁFICO DE TRÁFEGO DE DISCO

Este gráfico exibe a taxa de leitura e de escrita de dados no dispositivo de armazenamento sendo monitorado. Por “taxa” entende-se a quantidade de informação lida ou gravada em um determinado intervalo de tempo.

Um dos servidores monitorados durante a validação atendia requisições de uma aplicação WEB, cujo horário de pico era entre 19:00 e 22:00. Fazendo uso de uma amostra de 7 dias de métricas coletadas, pôde-se identificar este padrão de consumo, como pode ser visto na Figura 83.

Figura 83 - Padrão de tráfego de disco durante 7 dias



Fonte: Autoria própria

Na posse dessa informação, para fins de disparo de alerta foi configurado um limite, de forma que uma notificação fosse enviada ao usuário no momento em que a métrica se mantenha acima do valor configurado. A Figura 84 ilustra tal notificação, enviada em um momento de pico de utilização.

Figura 84 - Alerta referente ao tráfego de disco

Fonte: Autoria própria

Como é possível visualizar no alerta, a taxa de leitura do servidor monitorado se manteve constantemente acima do limite estabelecido (indicado pela linha horizontal vermelha).

Durante a validação da solução, pôde-se ainda constatar a relação entre o tráfego de disco e outras métricas monitoradas: o número de operações em disco comumente varia conforme as taxas de leitura e escrita. Além disso, o gráfico de utilização de disco obviamente demonstra um acréscimo na utilização em momentos de escrita, e o *disk caching* do sistema operacional se torna evidente: em momentos de grande tráfego de disco, os dados trafegados

são parcialmente alocados em memória RAM para acesso otimizado, seguindo uma política de despejo semelhante ao algoritmo LRU2Q (KERNEL, 2019).

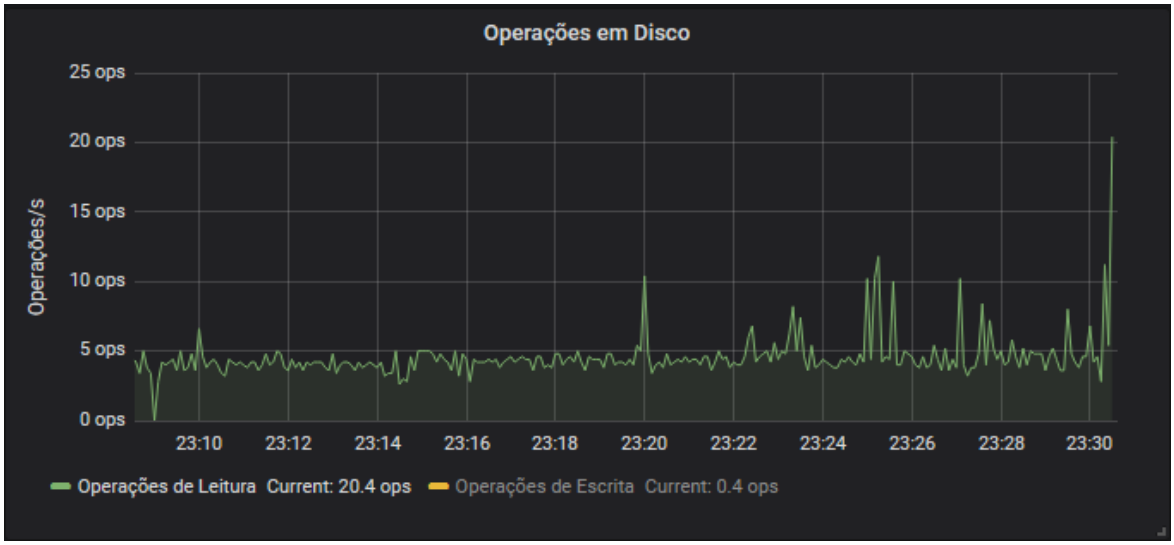
7.8 GRÁFICO DE NÚMERO DE OPERAÇÕES EM DISCO

O gráfico de operações em disco indica o número de operações de leitura e escrita executadas no dispositivo de armazenamento. Este gráfico tende a acompanhar as variações do tráfego de disco, porém há ressalvas dependendo do tamanho e quantidade de arquivos lidos e/ou escritos.

No contexto da validação, este gráfico é notadamente útil na análise de servidores cuja aplicação tende a executar muitos acessos randômicos no dispositivo, ou seja, o acesso a blocos não contíguos. Bancos de dados são um dos casos mais comuns, embora o emprego de *buffer pools* tenha grande influência nesta métrica (visto que evita executar operações em disco para buscar dados presentes em memória RAM).

Nenhuma das máquinas monitoradas durante a coleta de métricas atingiu um número suficientemente elevado de operações em disco a ponto de causar interrupções no fornecimento de serviços. Isto se deve em grande parte à utilização de dispositivos de armazenamento rápidos por parte da organização, além do fato de que muitas das tarefas que comumente executam um grande número de operações são otimizadas para não consumirem muito tempo de processamento e acesso em disco. A Figura 85 ilustra um destes casos.

Figura 85 - Gráfico de operações em disco durante varredura de antivírus



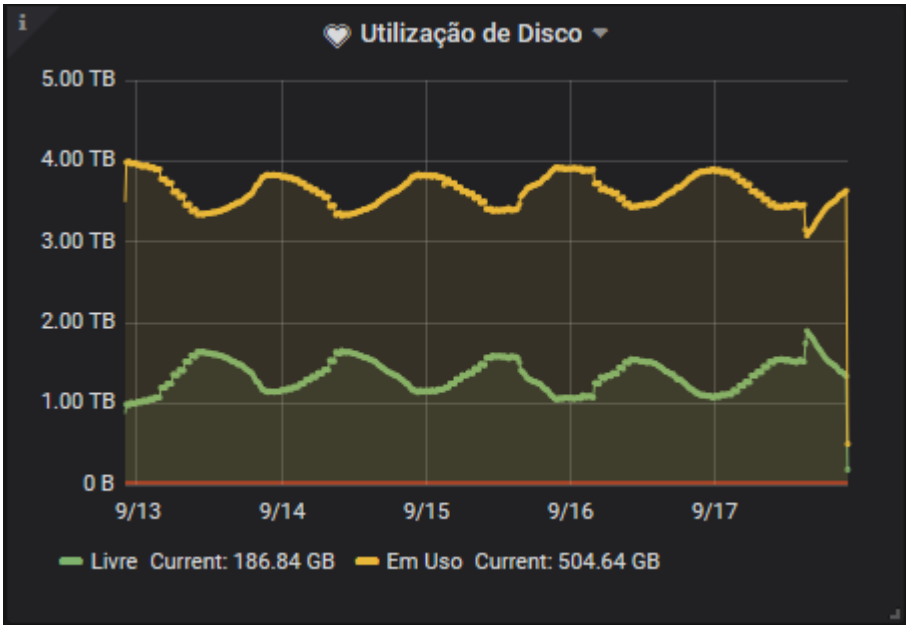
Fonte: Autoria própria

A figura 85 refere-se ao momento em que uma tarefa de varredura de *malware* é executada em um dos servidores. O próprio *software* antivírus, o qual é projetado única e exclusivamente para sistemas operacionais de servidores, já possui por padrão uma limitação na quantidade de operações. O objetivo é evitar que as funções do servidor sejam prejudicadas no momento de varreduras.

7.9 GRÁFICO DE UTILIZAÇÃO DE DISCO

Este gráfico indica a capacidade livre e ocupada do dispositivo de armazenamento, considerando a soma de todas as partições presentes e acessíveis pelo agente. Em situações de uso normais, é o gráfico com variações mais lentas e gradativas, considerando as amostras coletadas. A figura 86 exemplifica este comportamento.

Figura 86 - Variações de utilização de disco



Fonte: Autoria própria

No caso do servidor ao qual se refere a figura 86, a execução de tarefas de rotina que fazem a limpeza de arquivos de *cache* (em um intervalo fixo de tempo) faz com que a utilização de disco diminua durante estes intervalos.

A organização fez uso da solução proposta para definir o tempo de intervalo para a execução desta tarefa, levando em consideração a capacidade do servidor. Para isso, foi configurada uma regra de alerta, a qual não só avisava, como deixava registrado o momento exato em que o servidor atingia a ocupação máxima estipulada. Fazendo a análise da série

temporal coletada, pôde-se calcular um intervalo válido de limpeza de *cache* em disco que evite a utilização total, ao mesmo tempo que faz uso de toda a capacidade útil do servidor. A Figura 87 demonstra o alerta enviado.

Figura 87 - Alerta de utilização de disco

Fonte: Autoria própria

No contexto de validação, ainda foi possível fazer uso do gráfico e das métricas coletadas para o provisionamento mais eficiente de disco. Como as séries temporais indicam uma relação entre tempo e ocupação, pode-se estabelecer uma razão a ser utilizada para o provisionamento de máquinas semelhantes.

7.10 CONSIDERAÇÕES FINAIS

A solução atendeu as necessidades da organização no que tange o monitoramento de servidores e o envio de alertas, fazendo-o de forma compatível com a alocação de recursos financeiros e tecnológicos que se pode exigir de uma micro ou pequena organização. A utilização da solução possibilitou a capacidade de tornar a alocação de recursos computacionais

mais eficiente, além de ser empregada também na identificação, análise e resolução de problemas nas aplicações em execução nos ambientes monitorados.

Pode-se considerar ainda que a solução agregou valor ao conjunto de ferramentas utilizados pela organização em seu processo produtivo.

8 CONCLUSÃO

A crescente exigência de garantia de qualidade no fornecimento de serviços e interoperabilidade de aplicações fez surgir uma necessidade de acesso e conhecimento do estado de serviços e recursos computacionais.

Para atender esta necessidade, uma ampla gama de ferramentas de diversas naturezas surgiu, com destaque para as ferramentas de propósito geral para o monitoramento de servidores e ferramentas específicas para plataformas IaaS, as quais tem foco tanto no provisionamento de recursos como no planejamento de alocação, análise e mensuração de níveis de serviço.

Tais soluções, porém, não são acessíveis para uma grande parcela das organizações: ferramentas que atendem a um perfil de organização não necessariamente atende a outro, seja financeiramente e/ou tecnologicamente. A busca de soluções que atendam as necessidades de monitoramento e alerta de servidores em pequenas organizações foi o que motivou este trabalho.

8.1 DESENVOLVIMENTO DO TRABALHO

A primeira etapa deste trabalho teve dois objetivos básicos: conhecer a estrutura e funções comuns de ferramentas de monitoramento de métricas existentes no mercado, bem como fazer um levantamento e estudo de métricas relevantes a serem coletadas em servidores Linux. Para isso, foram analisados os modelos gerente-agente típicos deste tipo de ferramenta, bem como as diferentes formas de integração entre ambas as partes, e foi feito um estudo teórico sobre métricas de baixo nível e seu papel no gerenciamento de recursos computacionais.

Na etapa seguinte, conhecendo as funções e necessidades a serem atendidas pela solução, formalizou-se a intenção de empregar três categorias de ferramentas distintas (coleta, armazenamento e visualização/alertas), as quais viriam a cumprir as necessidades do projeto. Para isso, foram analisadas três ferramentas *open-source* de cada categoria, e fazendo a análise de critérios estabelecidos, selecionou-se uma ferramenta de cada tipo.

Embora não tenha sido difícil encontrar ferramentas, um dos fatores que dificultaram a seleção das mesmas foi a falta de publicações científicas com análises confiáveis, principalmente no que tange *softwares* coletores *open-source*.

Durante a seleção de ferramentas, houve a necessidade da análise de critérios fundamentais para a viabilidade da implementação, como a retrocompatibilidade entre ferramentas e a capacidade de customização de parâmetros importantes, como granularidade e

canais de comunicação. Além disso, critérios como a licença e compatibilidade com o ambiente de implementação também tiveram grande importância.

Após selecionadas as ferramentas, a proposta de solução foi formulada. Foram documentados requisitos técnicos, funcionais e não-funcionais, os quais viriam a ser cumpridos na implementação do projeto.

Buscando seguir o modelo de arquitetura de soluções já existentes no mercado e considerando os casos de uso criados, a arquitetura da solução foi planejada. Para o esclarecimento de dúvidas referentes às ferramentas utilizadas, alguns testes técnicos já foram feitos antes mesmo da implementação no ambiente de validação.

O desenvolvimento da solução proposta ocorreu no ambiente de validação: uma pequena empresa de desenvolvimento de *softwares* da região nordeste do Rio Grande do Sul. O ambiente que viria a hospedar as ferramentas foi preparado, de forma a garantir que a solução não interferisse com os serviços da organização. Fazendo uso de virtualização, foram provisionados os recursos definidos na etapa de proposta de solução, e após a parametrização e ativação das ferramentas, pôde-se estabelecer um fluxo de comunicação entre as mesmas.

Durante a implementação, a organização fez um questionamento referente à abertura de portas de rede na infraestrutura. Foi necessário explicar a natureza do tráfego, e tal preocupação por parte da organização motivou a busca por uma solução de autenticação entre as ferramentas de coleta e armazenamento, a qual foi implementada.

Visto que um maior número de métricas garante uma melhor noção dos recursos e da utilização da solução, a etapa de validação iniciou com a coleta de métricas de servidores. Houve uma dificuldade na aplicação da política de retenção de métricas definida, ocorrida pelo desconhecimento do conceito e da forma de aplicação. Após uma pesquisa mais profunda, foi possível executar os procedimentos necessários. Devido a esta dificuldade, porém, aproximadamente uma semana de métricas coletadas foram perdidas devido à necessidade de recriação do modelo de dados para a aplicação da política de retenção surtir efeito.

Ao mesmo tempo em que a solução esteve coletando métricas, os casos de testes foram executados com o objetivo de garantir que todas as funcionalidades propostas estavam funcionais e de acordo com o planejamento inicial. Também foi feita uma análise de utilização de recursos, como uma forma de assegurar a viabilidade da solução independente da quantidade de métricas ou consultas.

Durante o período de validação, a ferramenta desenvolvida foi empregada pela organização no monitoramento de métricas, e se mostrou útil tanto no gerenciamento da infraestrutura como no desenvolvimento e correções de *bugs* de aplicações em execução.

8.2 CONTRIBUIÇÕES

A solução implementada neste projeto foi projetada com o objetivo de garantir a pequenas empresas a capacidade de monitorar o estado e a utilização de recursos computacionais em servidores Linux. Esta necessidade, que antes era de grandes organizações que dispunham de muitos recursos financeiros e tecnológicos, hoje já é uma necessidade de qualquer organização que busque oferecer um nível de qualidade de serviço competitivo.

Orientando-se pelos quatro objetivos específicos deste trabalho (definir métricas, analisar ferramentas, projetar, e implementar a solução), a integração de ferramentas para a coleta de métricas em servidores Linux foi desenvolvida e validada no contexto planejado. Como descrito no capítulo 6 (Testes e resultados), todas as funcionalidades previamente planejadas, as quais viriam a suportar o processo de coleta, armazenamento, visualização e disparo de alertas de métricas foram implementadas com sucesso.

Durante o período de validação da solução, a mesma foi utilizada por 2 usuários da organização. Tais usuários conseguiram, através da visualização das séries temporais, identificar incidentes recorrentes que antes sequer eram conhecidos. Não apenas isso, como foi possível constatar problemas nas aplicações em execução nos ambientes monitorados, principalmente no que tange a parametrização de uso de recursos.

Porém, a maior contribuição da solução para a organização foi o aumento da eficiência na utilização de recursos durante a criação de ambientes (*deploy*). Foi possível identificar a utilização de recursos por servidores, aplicações e dependências em situações reais, o que pôde ser documentado e utilizado em procedimentos futuros.

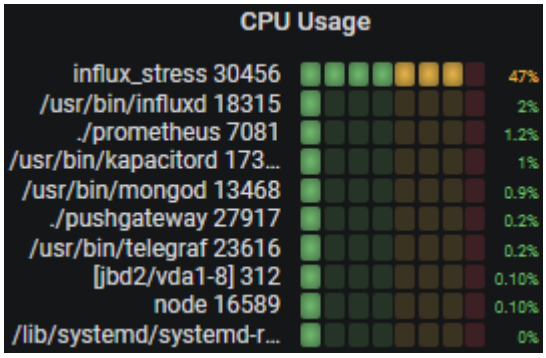
Referente à usabilidade, os usuários da ferramenta relataram que a interface é amigável o suficiente para que o aprendizado ocorra sem a necessidade de um treinamento, e a utilização de recursos pela solução se manteve dentro dos valores esperados. Isso, somado ao fato de que os paineis e alertas implementados tiveram um impacto positivo para a organização (como foi documentado no capítulo 7), permite concluir que a solução proposta atingiu os objetivos propostos no início do trabalho.

8.3 TRABALHOS FUTUROS

Algumas melhorias podem ser realizadas futuramente na implementação da ferramenta. Durante a utilização da mesma, foi possível identificar a necessidade da coleta da lista de

processos em execução no ambiente monitorado e seus respectivos estados, o que pode permitir identificar com maior exatidão a causa das variações de utilização de recursos, bem como garante uma visão mais clara do comportamento das aplicações no servidor. A Figura 88 demonstra uma forma de implementar este painel.

Figura 88 - Exemplo de painel de lista de processos



Fonte: Autoria própria

Também, identificou-se que a criação de um *script* para a instalação, parametrização e configuração do serviço de coleta de métricas (agentes) pode agilizar o processo de implantação da solução em novos servidores, além de ser algo simples de colocar em prática.

REFERÊNCIAS BIBLIOGRÁFICAS

ACETO, Giuseppe; BOTTA, Alessio; DONATO, Walter de; PESCAPÈ, Antonio. Cloud Monitoring: A survey. **Computer Networks**, [s. l.], v. 57, n. 9, p. 2093-2115, jun. 2013. DOI: dx.doi.org/10.1016/j.comnet.2013.04.001. Disponível em: <https://www.academia.edu/20154217/Cloud_monitoring_A_survey>. Acesso em 01 abr. 2019.

AMAZON. **Using Amazon CloudWatch Alarms**. Disponível em: <https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/AlarmThatSendsEmail.html>. Acesso em: 07 abr. 2019.

APARECIDO, Wanderson. **Kibana: Incrível ferramenta para você analisar suas aplicações, bancos e servidores**. 23 nov. 2017. Disponível em: <https://medium.com/@wanderson.coord/kibana-incr%C3%ADvel-ferramenta-para-voc%C3%AA-analisar-suas-aplica%C3%A7%C3%B5es-bancos-e-servidores-6824d825b3da>. Acesso em: 19 abr. 2019.

ARSENAULT, Cody. **Running High Traffic Websites - 8 Things to Consider**. 6 jun. 2017. Disponível em: <https://www.keycdn.com/blog/high-traffic>. Acesso em: 12 abr. 2019.

AZURE. **Overview of alerts in Microsoft Azure**. Disponível em: <https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-overview>. Acesso em: 07 abr. 2019.

BONCEA, Radu; ZAMFIROIU, Alin; BACIVAROV, Ioan. A scalable architecture for automated monitoring of microservices. **Economy Informatics**, [s. l.], v. 18, n. 1, p. 17, 2018. ISSN 1582-7941 versão online. Disponível em: <http://www.economyinformatics.ase.ro/content/EN18/02%20-%20boncea,%20zamfiroiu,%20bacivarov.pdf>. Acesso em 11 abr. 2019.

BORINI, Guilherme. **Gastos com TI representam 7,7% da receita das empresas brasileiras, diz FGV**. 2018. Disponível em:

<https://computerworld.com.br/2018/04/19/gastos-com-ti-representam-77-da-receita-das-empresas-brasileiras-diz-fgv/>. Acesso em: 10 mar. 2019.

BORINI, Guilherme. **TI deve crescer 10% em 2019 no Brasil. Saiba onde estarão os investimentos**, 2019. Disponível em: <https://computerworld.com.br/2019/02/06/ti-deve-crescer-10-em-2019-no-brasil-saiba-onde-estarao-os-investimentos/> Acesso em: 10 mar. 2019.

BOVET, Daniel Pierre; CESATI, Marco. **Understanding the Linux Kernel**. 2 ed.; Sebastopol; O'Reilly Media, Inc.; 2003.

BROWN, Ben. **Facebook's Catastrophic Blackout Could Cost $90 Million in List Revenue**. Disponível em: <https://www.ccn.com/facebooks-blackout-90-million-lost-revenue>. Acesso em: 17 mar. 2019.

CAEN, François; NEGUS, Christopher. **Fedora Linux Toolbox: 1000+ Commands for Fedora, CentOS and Red Hat Power Users**. Indianapolis; Wiley Publishing, 2008.

CARTER, Eric. **5 examples of Prometheus monitoring success**. 12 set. 2018. Disponível em: <https://opensource.com/article/18/9/prometheus-operational-advantage>. Acesso em: 18 abr. 2019.

CHOUDHURY, Diptanu Gon; PERRETT, Timothy. Designing Cluster Schedulers for Internet-Scale Services. **Communications of the ACM**, [s. l.], v. 16, n. 1, p. 01-30, jan. 2018. DOI: https://doi.org/10.1145/3194653.3199609. Disponível em: <https://dl.acm.org/ft_gateway.cfm?id=3199609&ftid=1953553&dwn=1>. Acesso em 05 abr. 2019.

COLLECTD. **Collectd - The system statistics collection daemon**. Disponível em: <https://collectd.org/>. Acesso em: 14 abr. 2019.

COLLECTD. **FQDNLookup - collectd Wiki**. 6 ago. 2014. Disponível em: <https://collectd.org/wiki/index.php/FQDNLookup>. Acesso em 14 abr. 2019.

COMER, Douglas E. **Redes de Computadores e a Internet**. 6 ed. New Jersey. Bookman; 2016.

COMPUTERWORLD. **80% de dados e sistemas ainda não foram para nuvem**. Disponível em: <https://computerworld.com.br/2019/02/15/80-de-dados-e-sistemas-ainda-nao-foram-para-nuvem/>. Acesso em: 22 mar. 2019.

CONVERGÊNCIA DIGITAL. **Empresas gastaram, e bem, com infraestrutura de TI para nuvem pública e privada**. Disponível em: <http://www.convergenciadigital.com.br/cgi/cgilua.exe/sys/start.htm?UserActiveTemplate=site&infoid=49870&sid=97>. Acesso em: 22 mar. 2019.

CORREIO BRAZILIENSE. **Profissionais de TI apontam falta de recursos como maior obstáculo do setor**. Disponível em: <https://www.correiobraziliense.com.br/app/noticia/tecnologia/2015/03/28/interna_tecnologia,477327/profissionais-de-ti-apontam-falta-de-recursos-como-maior-obstaculo-do.shtml>. Acesso em: 22 mar. 2019.

DALE, Nell, LEWIS, John. **Ciência da Computação**, 4 ed. Rio de Janeiro, RJ; LTC, 2011.

DAVIS, Chris. **The Architecture of Open Source Applications: Graphite**. 07 jul. 2012. Disponível em: <https://www.aosabook.org/en/graphite.html>. Acesso em: 18 abr. 2019.

DB-ENGINES. **Popularity ranking of time series DBMS**. Disponível em: <https://db-engines.com/en/ranking/time+series+dbms>. abr. 2019.

DIEESE. **Anuário do Trabalho nos Pequenos Negócios**, 2018. Disponível em: <https://www.dieese.org.br/anuario/2018/anuarioDosTrabalhadoresPequenosNegocios.html>. Acesso em: 9 mar. 2019.

ELASTIC. **Graphite Input Plugin | Logstash Reference [7.0] | Elastic**. Disponível em: <https://www.elastic.co/guide/en/logstash/current/plugins-inputs-graphite.html>. Acesso em 19 abr. 2019.

ELASTIC. **Kibana: Explore, Visualize, Discover Data | Elastic**. Disponível em: <https://www.elastic.co/products/kibana>. Acesso em: 19 abr. 2019.

ELLINGWOOD, Justin. **An Introduction to Metrics, Monitoring and Alerting**. Disponível em: <https://www.digitalocean.com/community/tutorials/an-introduction-to-metrics-monitoring-and-alerting>. Acesso em: 08 abr. 2019.

ELLIS, Alex. **Monitor your applications with Prometheus**. 19 mar. 2017. Disponível em: <https://blog.alexellis.io/prometheus-monitoring/>. Acesso em: 18 abr. 2019.

ENVIROMON. **Calculating the Cost of Downtime for Data Centers and Businesses**. Disponível em: <https://www.enviromon.net/calculating-downtime-cost-data-centers-businesses/>. Acesso em: 17 mar. 2019.

EVEO. **3 aspectos importantes sobre o uso do IaaS como suporte de negócios**, 2018. Disponível em: <https://www.eveo.com.br/blog/aspectos-sobre-iaas/>. Acesso em: 9 mar. 2019.

FARMER, Katy. **How Predefined Dashboards in InfluxData's Chronograf Make Metrics Simple**. 17 dez. 2018. Disponível em: <https://dzone.com/articles/how-predefined-dashboards-in-influxdatas-chronogra>. Acesso em: 19 abr. 2019.

FATEMA, Kaniz et al. A survey of Cloud monitoring tools: Taxonomy, capabilities and objectives. **Journal of Parallel and Distributed Computing**, [s. l.], v. 74, n. 10, p. 2918-2933, out. 2014. DOI: dx.doi.org/10.1016/j.jpdc.2014.06.007. Disponível em: <http://romisatriawahono.net/lecture/rm/survey/network%20security/Fatema%20-%20Cloud%20Monitoring%20Tools%20-%202014.pdf>. Acesso em 02 abr. 2019.

GETTI. **Cloud Server x Servidor on Premise: entenda as diferenças**. Disponível em: <https://getti.net.br/cloud-server-x-servidor-on-premise-entenda-as-diferencas/>. Acesso em: 22 mar. 2019.

GITE, Vivek. **Linux Check Memory Usage**. 7 jan. 2013. Disponível em: <https://www.cyberciti.biz/faq/linux-check-memory-usage/>. Acesso em: 12 abr. 2019.

GITHUB. **Collectd/collectd Repository**. Out. 2018. Disponível em: <https://github.com/collectd/collectd>. Acesso em: 14 abr. 2019.

GITHUB. **CymaticLabs/InfluxDBStudio: InfluxDB Studio is a UI management tool for the InfluxDB time series database**. 13 out. 2017. Disponível em: <https://github.com/CymaticLabs/InfluxDBStudio>. Acesso em: 05 mai. 2019.

GITHUB. **Influxdata/influxql repository**. Disponível em: <https://github.com/influxdata/influxql>. Acesso em: 16 abr. 2019.

GITHUB. **InfluxData/Telegraf repository**. Disponível em: <https://github.com/influxdata/telegraf>. Acesso em: 14 mar. 2019.

GITHUB. **Netdata/netdata repository**. Disponível em: <https://github.com/netdata/netdata>. Acesso em: 14 mar. 2019.

GNU. **The GNU General Public License v3.0**. Disponível em: <https://www.gnu.org/licenses/gpl.html>. Acesso em: 31 mar. 2019.

GRAFANA. **Grafana Documentation**. Disponível em: <https://grafana.com/docs/>. Acesso em: 19 abr. 2019.

GRAPHITE. **Graphite - The Architecture in a Nutshell**. Disponível em: <https://graphiteapp.org/#integrations>. Acesso em: 18 abr. 2019.

GRAPHITE DOCS. **Installing Graphite - Graphite 1.1.5 documentation**. Disponível em: <https://graphite.readthedocs.io/en/latest/install.html>. Acesso em: 18 abr. 2019.

GREGG, B. **Systems Performance: Enterprise and the Cloud**. New Jersey, Prentice Hall; 7 de out. de 2013.

INDORE, Swapnil. **Get Disk Space Email Alerts from your Linux Servers**. 23 fev. 2015. Disponível em: <https://www.linux.com/blog/get-disk-space-email-alerts-your-linux-servers>. Acesso em: 10 abr. 2019.

INFLUXDATA. **Chronograf 1.7 Documentation | InfluxData Documentation**. Disponível em: <https://docs.influxdata.com/chronograf/v1.7/>. Acesso em: 19 abr. 2019.

INFLUXDATA. **Chronograf | Complete Interface for the InfluxData Platform**. Disponível em: <https://www.influxdata.com/time-series-platform/chronograf/>. Acesso em: 19 abr. 2019.

INFLUXDATA. **InfluxDB compared to SQL databases**. Disponível em: https://docs.influxdata.com/influxdb/v1.7/concepts/crosswalk/. Acesso em 18 abr. 2019.

INFLUXDATA. **Telefraf 1.10 documentation**. Disponível em: <https://docs.influxdata.com/telegraf/v1.10/>. Acesso em: 14 mar. 2019.

JONES, Don. **The Definitive Guide To SQL Server Performance Optimization**. [s. l.], Realtime Publishers; 5 de out. de 2014.

JONES, Martino. **Monitor Linux With Netdata**. 20 jun. 2016. Disponível em: <https://fedoramagazine.org/monitor-linux-netdata/>. Acesso em: 19 abr. 2019.

JUNIOR, Miguel Garcia; SANTANA, Rodrigo. P**rocesso de gerenciamento de continuidade do serviço de TI**, 2012. Disponível em: <https://www.diegomacedo.com.br/processo-de-gerenciamento-da-continuidade-do-servico-de-ti/>. Acesso em: 9 mar, 2019.

KAMENOV, Drago. **Key Linux Performance Metrics**. 7 set. 2007. Disponível em: <https://www.monitis.com/blog/key-linux-performance-metrics/>. Acesso em: 07 abr. 2019.

KERNEL. **Page Frame Reclamation**. Disponível em <https://www.kernel.org/doc/gorman/html/understand/understand013.html>. Acesso em: 16 set. 2019.

KERNEL. **The Linux Kernel Archives - Releases**. 26 aug. 2019. Disponível em <https://www.kernel.org/category/releases.html>. Acesso em: 08 set. 2019.

KERNEL. **The Linux kernel user's and administrator's guide: Memory Management**. Disponível em: <https://www.kernel.org/doc/html/latest/admin-guide/mm/concepts.html#oom-killer>. Acesso em: 12 abr. 2019.

KHAN, Janshair. **Monitoring With Prometheus**. 27 fev. 2018. Disponível em: https://dzone.com/articles/monitoring-with-prometheus. Acesso em: 18 abr. 2019.

KOZLOWICZ, Joe. **Know Your Storage Constraints: IOPS and Throughput**. 28 set. 2017. Disponível em: <https://www.greenhousedata.com/blog/know-your-storage-constraints-iops-and-throughput>. Acesso em: 10 abr. 2019.

KREEFTMEIJER, Jeff. **Understanding CPU statistics**. 6 mar. 2018. Disponível em: <https://blog.appsignal.com/2018/03/06/understanding-cpu-statistics.html>. Acesso em: 12 abr. 2019.

LIU, Rui; YUAN, Jun. **Benchmark Time Series Database with IoT: DB-Benchmark for IoT Scenarios**. 1 fev. 2019. Disponível em: <https://arxiv.org/abs/1901.08304>. Acesso em: 14 abr. 2019.

MACHADO, Francis B., MAIA, Luiz Paulo. **Arquitetura de Sistemas Operacionais**. 5 ed. Rio de Janeiro, RJ; LTC; 2017.

MAN7. **Free(1) - Linux Manual Page**. Disponível em: <http://man7.org/linux/man-pages/man1/free.1.html>. Acesso em: 07 abr. 2019.

MAN7. **Proc(5) - Linux Programmer's Manual**. Disponível em: <http://man7.org/linux/man-pages/man5/proc.5.html>. Acesso em: 07 abr. 2019.

MAN7. **Top(1) - Linux Manual Page**. Disponível em: <http://man7.org/linux/man-pages/man1/top.1.html>. Acesso em: 07 abr. 2019.

MOLINA, Alberto. **Rsyslog, journal or both?**. 30 dez. 2017. Disponível em: <https://albertomolina.wordpress.com/2017/12/30/rsyslog-journal-or-both/>. Acesso em: 10 abr. 2019.

MOTODATA. **Web Server Monitoring - Why It's Very Important for Organizations**. Disponível em: <https://www.motadata.com/blog/web-server-monitoring-top-4-reasons-why-organization-needs-it/>. Acesso em: 17 mar. 2019.

NAGIOS. **History of Nagios**. Disponível em: <https://www.nagios.org/about/history/>. Acesso em: 22 mar. 2019.

NAGIOS. **Nagios Core**. Disponível em: <https://www.nagios.org/projects/nagios-core/>. Acesso em: 22 mar. 2019.

NAGIOS. **What is Nagios?**. Disponível em: <https://www.nagios.org/about/>. Acesso em: 17 mar. 2019.

NETADMINTOOLS. **Best Nagios Alternatives**. Disponível em: <https://www.netadmintools.com/nagios-alternatives>. Acesso em: 17 mar. 2019.

NETDATA. **Netdata Documentation**. Disponível em: <https://docs.netdata.cloud/>. Acesso em: 14 mar. 2019.

NURGALIEV, Ildar; KARAVAKIS, Edward; AIMAR, Alberto. Kibana, Grafana and Zeppelin on Monitoring data. **CERN openlab Summer Student Report**, [s.l.], 2016.

OPSDASH. **Understanding CPU Usage in Linux**. Disponível em: <https://www.opsdash.com/blog/cpu-usage-linux.html>. Acesso em: 07 abr. 2019.

ORACLE. **Database SQL Tuning Guide**. Disponível em: <https://docs.oracle.com/database/121/TGSQL/tgsql_intro.htm#TGSQL114>. Acesso em: 22 mar. 2019.

ÖZMEN, Ahmet. An entropy-based algorithm for data elimination in time-driven software instrumentation. **Journal of Systems and Software**, [s. l.], v. 82, n. 5, p. 907-913, mai. 2009. DOI: https://doi.org/10.1109/DSN.2017.39. Disponível em: <https://www.researchgate.net/publication/316145415_Entropy-Based_Security_Analytics_Measurements_from_a_Critical_Information_System>. Acesso em 6 abr. 2019.

PROFISSIONAIS TI. **A realidade de ambientes de TI em Micro e Pequenas Empresas (MPE)**. Disponível em: <https://www.profissionaisti.com.br/2014/02/a-realidade-de-ambientes-de-ti-em-micro-e-pequenas-empresas-mpe/>. Acesso em: 30 mar. 2019.

PROMETHEUS. **Overview | Prometheus**. Disponível em: <https://prometheus.io/docs/introduction/overview/>. Acesso em: 18 abr. 2019.

PROMETHEUS. **HTTP API | Prometheus**. Disponível em: <https://prometheus.io/docs/prometheus/latest/querying/api/>. Acesso em 03 mai. 2019.

RAMOS, J. **Guia prático do servidor Linux: Administração Linux para iniciantes**. São Paulo; Casa do Código, 3 de ago. de 2018.

RIAZ, Zainab et al. BIM and sensor-based data management system for construction safety monitoring. **Journal of Engineering Design and Technology**, [s. l.], v. 15, n. 6, p. 749, dez. 2017. DOI: https://doi.org/10.1108/JEDT-03-2017-0017. Disponível em: <https://www.emeraldinsight.com/doi/abs/10.1108/JEDT-03-2017-0017>. Acesso em 10 abr. 2019.

SCHAEDEL, Margo. **Simplifying InfluxDB: Retention Policy Best Practices**. 22 jun. 2018. Disponível em: <https://dzone.com/articles/simplifying-influxdb-retention-policy-best-practic>. Acesso em 11 abr. 2019.

SIMMONS, David. **How to: Building Flux Queries in Chronograf**. 19 nov. 2018. Disponível em: <https://dzone.com/articles/how-to-building-flux-queries-in-chronograf>. Acesso em: 19 abr. 2019.

SIMS, Gary. **All about Linux swap space**. 2007. Disponível em: <https://www.linux.com/news/all-about-linux-swap-space>. Acesso em: 07 abr. 2019.

SOFTWARE ONE. **Managing and Understanding On-Premises and Cloud Spend**. Disponível em: <https://www.softwareone.com/on-premises-and-cloud-spend-survey/>. Acesso em: 22 mar. 2019.

SOL, Tony. **PHP 5.6 vs PHP 7 Performance Comparison**. Disponível em: <https://gbksoft.com/blog/php-5-vs-php-7-performance-comparison/>. Acesso em: 22 mar. 2019.

SOUZA, Ribamar Ferreira de. **Servidor Web com foco em VPS**. Fortaleza: Clube de Autores, 2014.

STACKSHARE. **Best5 Monitoring Tools Software - 2019 Reviews of the Most Popular | Stackshare**. Disponível em: <https://stackshare.io/monitoring-tools>. Acesso em 03 mai. 2019.

STENICO, Jeferson Wilian de Godoy; LING, Lee Luan. Network Traffic Monitoring and Analysis. In: Al-Sakib Khan Pathan. (coord.). **The State of the Art in Intrusion Prevention and Detection**. [s. l.]: CRC Press, 2014, p. 23-46.

SYKES, Andy. **Stop using Nagios (so it can die peacefully)**. Disponível em: <https://www.slideshare.net/superdupersheep/stop-using-nagios-so-it-can-die-peacefully/>. Acesso em: 22 mar. 2019.

TAHERIZADEH, Salman et al. Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review. **Journal of Systems and Software**, [s. l.], v. 136, p. 23, fev 2018. DOI: https://doi.org/10.1016/j.jss.2017.10.033. Disponível em:

<https://www.sciencedirect.com/science/article/pii/S016412121730256X>. Acesso em 5 abr. 2019.

TECHGENIX. **Top 10 open-source application monitoring tools**. Disponível em: <http://techgenix.com/open-source-application-monitoring-tools/>. Acesso em 03 mai. 2019.

TELEGRAM. **Bots: An Introduction for Developers**. Disponível em: <https://core.telegram.org/bots>. Acesso em 10 ago. 2019.

THE LINUX KERNEL ARCHIVES. **I/O Statistics Fields**. Disponível em: <https://www.kernel.org/doc/Documentation/iostats.txt>. Acesso em: 11 abr. 2019.

TINARI, George. **Google's 5-minute outage means $545,000 revenue loss, 40% drop in global website traffic**. Disponível em: <https://www.neowin.net/news/googles-5-minute-outage-means-545000-revenue-loss-40-drop-in-global-website-traffic>. Acesso em: 17 mar. 2019.

TURNBULL, J. **The Art of Monitoring**. [s. l.]: Amazon Digital Services LLC; 8 de jun. de 2016.

W2 COOK. **OS Market Share and Usage Trends.** 2016. Disponível em: <https://web.archive.org/web/20160407103919/http://www.w3cook.com/os/summary>. Acesso em: 9 mar. 2019.

W3TECHS. **Usage of Operating Systems for Websites**. 2013. Disponível em: <https://w3techs.com/technologies/overview/operating_system/all>. Acesso em: 10 mar. 2019.

WARD, Jonathan Stuart; BARKER, Adam. Observing the clouds: a survey and taxonomy of cloud monitoring. **Journal of Cloud Computing.** [s. l.], v. 3, n. 24, p. 01-24, dez. 2014. DOI: https://doi.org/10.1186/s13677-014-0024-2. Disponível em: <https://link.springer.com/article/10.1186/s13677-014-0024-2>. Acesso em 4 abr. 2019.

ZABBIX. **Open-source Enterprise Monitoring with Zabbix**. Disponível em: <https://web.archive.org/web/20120226220044/http://www.netways.de/uploads/media/Alexei_Vladishev_Open_Source_Monitoring_with_Zabbix.pdf>. Acesso em: 17 mar. 2019.

ZABBIX. **Zabbix Documentation 4.0**. Disponível em: <https://www.zabbix.com/documentation/4.0/manual/installation/requirements>. Acesso em: 22 mar. 2019.

ZAITSEV, Peter. **Prometheus 2 Times Series Storage Performance Analyses**. 26 set. 2018. Disponível em: <https://dzone.com/articles/prometheus-2-times-series-storage-performance-anal>. Acesso em: 18 abr. 2019.