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

PEDRO HENRIQUE MENEGAZZI

MONITORAMENTO DE APLICAÇÕES EM TEMPO REAL UTILIZANDO O TELEGRAM

CAXIAS DO SUL
2021

PEDRO HENRIQUE MENEGAZZI

MONITORAMENTO DE APLICAÇÕES EM TEMPO REAL UTILIZANDO O TELEGRAM

Trabalho de Conclusão de curso apresentado à Universidade de Caxias do Sul como requisito básico para a conclusão do curso de Tecnologias Digitais.

Orientadora: Profª. Drª. Elisa Boff

CAXIAS DO SUL
2021

PEDRO HENRIQUE MENEGAZZI

MONITORAMENTO DE APLICAÇÕES EM TEMPO REAL UTILIZANDO O TELEGRAM

Trabalho de Conclusão de curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel em Tecnologias Digitais 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. MSc. Alexandre Erasmo Krohn Nascimento
Universidade de Caxias do Sul – UCS

_____
Profª. Drª. Elisa Boff
Universidade de Caxias do Sul – UCS

Dedico este trabalho aos meus familiares, que nunca mediram esforços para que eu pudesse chegar até esta etapa da minha vida.

AGRADECIMENTOS

Agradeço primordialmente aos meus pais, Vitor e Bernardete, e a minha irmã Bruna, que através de muito trabalho e dedicação, me concederam a oportunidade de graduação em um curso que sempre sonhei.

Sou grato aos amigos e colegas de curso, por todo apoio e incentivo que foi me dado ao longo desta jornada. Agradeço também à minha orientadora, Profª. Drª. Elisa Boff pela paciência e entrega.

Ao meu colega de trabalho, Giovani de Souza Castro, que através do seu árduo conhecimento, pode me auxiliar em momentos de dúvida que apareceram ao longo deste trabalho.

RESUMO

Em empresas do ramo de tecnologia se vê a necessidade de monitorar ambientes relacionados à infraestrutura. Com a viabilidade de mecanismos de troca de mensagens, é possível ter a oportunidade de monitorar aplicações em tempo real. Por meio de *smartphones*, pode-se acompanhar métricas extremamente relevantes para a saúde de um ambiente monitorado. Com base nesse estudo, foi possível ter o entendimento sobre os mecanismos necessários para a monitoração e benefícios na utilização de ambientes Linux para empresas. Os *softwares* escolhidos foram o Zabbix (ferramenta de monitoramento), o Telegram (ferramenta de recebimento de alertas) e o Grafana (ferramenta especializada em gráficos para visualização). Os mesmos foram integrados juntamente a um banco de dados MariaDB, para que o resultado pudesse ser validado, onde foi possível a visualização de alertas frequentes de infraestrutura em tempo real, recebimento de alertas através do *smartphone* e resolução de alertas de forma automática, para que através da etapa de validação, fosse alcançada a eficiência ao monitorar ambientes e aplicações, permitindo que o usuário administrador pudesse ter um monitoramento proativo para desfrutar da alta disponibilidade.

Palavras-chave: Monitoramento de servidores, Alta disponibilidade, Zabbix.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

API	*Application Program Interface*
CPU	*Central Processing Unit*
BSD	*Berkeley Software Distribution*
CEO	*Chief Executive Officer*
FTP	*File Transfer Protocol*
GPLv3	*General Public License (Versão 3)*
HD	*Hard Drive*
HTTP	*Hypertext Transfer Protocol*
IMAP	*Internet Message Access Protocol*
POP	*Post Office Protocol*
RAM	*Random Access Memory*
SMTP	*Simple Mail Transfer Protocol*
SSD	*Solid State Drive*
SSH	*Secure Shell*
SaaS	*Software As A Service*
TCP	*Transmission Control Protocol*
VM	*Virtual Machine*
WWW	*World Wide Web*

SUMÁRIO

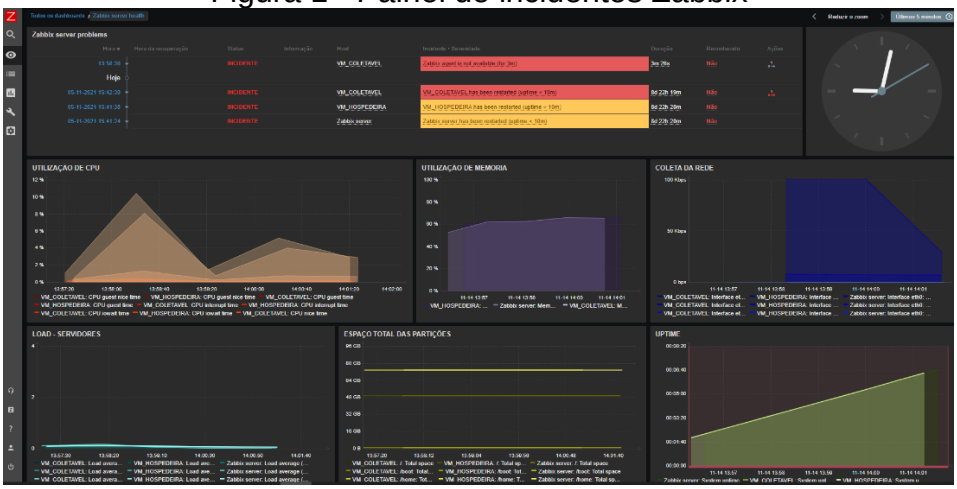
1 INTRODUÇÃO

A importância do monitoramento de aplicações se faz cada vez mais presente com o intuito de prevenção e segurança. Nos últimos anos, as aplicações se tornaram mais complexas, robustas e dinâmicas (PAL, 2017), em momentos onde os recursos visuais e ações humanas são limitados, os alertas em tempo real via plataformas de monitoramento são um ótimo auxilio para que a arquitetura computacional esteja sempre a favor da alta disponibilidade. Dessa forma, é possível atingir o objetivo de monitoramento constante, oferecendo diferentes maneiras de colaborar com a produtividade da aplicação, tanto quanto, com a satisfação do usuário.

Com o pensamento de tolerância a falhas, as soluções necessárias foram ficando cada vez mais complexas para suprirem as atividades humanas (DELL, 2020), com isso optou-se por aderir plataformas e ferramentas capazes de entregar alta disponibilidade e segurança para alertas que surjam e tenham a prioridade de serem vistos o mais rápido possível.

A ferramenta Zabbix, como pode-se ver na Figura 1, é uma solução de código aberto criada com o intuito de fazer o monitoramento de empresas via parâmetros flexíveis que se adaptam de acordo com cada aplicação, permitindo rápida resolução dos alertas e captação de recursos em tempo real (Zabbix Manual, 2020).

Figura 1 - Painel de incidentes Zabbix

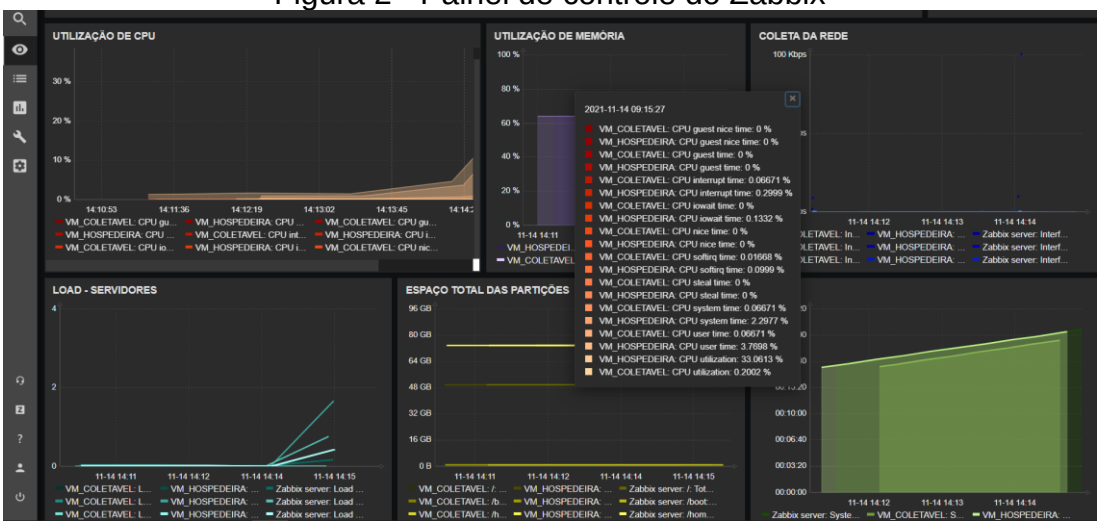


Fonte: Autoria própria

Além de ser projetado para ser executado em uma distribuição Linux, sua eficiência de monitoramento em uma estrutura está relacionada com a coleta de dados, alertas customizáveis, integração com *APIs*, entre outros, como mostra o

painel da Figura 2. Todos esses quesitos são capazes de proporcionar confiabilidade ao usuário.

Figura 2 - Painel de controle do Zabbix



Fonte: Autoria própria

Para que seja possível garantir a facilidade no recebimento de descobertas de alertas provindos do Zabbix, opta-se pelo o uso de uma plataforma de visualização capaz de receber os alertas em tempo real chamado Telegram.

Além de ser um serviço de mensagens instantâneas baseado na nuvem (*cloud-based*), o Telegram também providencia interfaces programáticas (*APIs*) para desenvolvedores independentes. Com isso os usuários podem receber notificações, fotos, arquivos entre outros.

Quando pesquisamos ferramentas alternativas para o envio instantâneo de mensagens podemos encontrar o WhatsApp. O mesmo refere-se a uma plataforma de comunicação de código fechado semelhante ao Telegram, possibilitando a comunicação entre a aplicação e o usuário (WhatsApp, 2021). Porém, para que o vínculo entre WhatsApp e Zabbix funcione, torna-se necessário o uso da biblioteca em Python chamada YowSup (Share Zabbix, 2021).

O presente trabalho apresenta um estudo sobre a fundamentação teórica de sistemas de monitoramento, servidores, métricas do sistema operacional para coleta de dados e a seleção das ferramentas para o propósito do desenvolvimento de um ambiente de monitoramento. Dessa forma, tornou-se possível o desenvolvimento um ambiente de monitoramento, capaz de propor e coletar alertas essenciais de infraestrutura, juntamente com a resolução automática de alertas sem a necessidade

de intervenção humana e envio de alertas em tempo real para o usuário final via *smartphone*, por meio do Telegram.

1.1 PROBLEMA E QUESTÃO DE PESQUISA

A gradatividade de alertas provindos de ferramentas capazes de fazerem as coletas de informações de servidores é algo que precisa ser notado e sempre utilizado a favor de uma empresa quando o objetivo é o monitoramento em tempo real. Porém, a partir do momento em que a aplicação não esteja configurada corretamente ou o crescimento de uma empresa seja notável, é humanamente inviável suprir a demanda de uma infraestrutura, analisando todos os alertas que um ambiente de monitoramento é capaz de produzir.

A constante melhoria das versões em ambientes de monitoramento é um fator importante para a integração ao usuário. Quando as aplicações são configuradas corretamente, as consequências são excelentes recursos capazes de serem utilizados por diversos setores empresariais, tanto quanto, desempenharem um papel verdadeiramente importante em uma infraestrutura, fazendo com que seja uma tecnologia ideal para planejamento e capacidade.

Com relação à questão de pesquisa, foi importante esclarecer o contexto. Além da fundamentação dos dados pesquisados, vivenciou-se a oportunidade de estar diretamente vinculado a uma empresa que utilizou uma plataforma de monitoramento. Durante o período, foi notável o crescimento de alertas e a incapacidade humana de resolução quando é preciso eficiência em grandes volumes de dados.

Questão de Pesquisa: O que é necessário para um ambiente de monitoramento fornecer alertas em tempo real para um *smartphone* via Telegram?

1.2 OBJETIVO GERAL

O objetivo geral deste trabalho foi um estudo sobre a fundamentação teórica das métricas mais relevantes de infraestrutura serem monitoradas em um ambiente com distribuição Linux, juntamente com a escolha das ferramentas capazes de disparar alertas em tempo real para o Telegram, tanto quanto, resolverem os alertas

sem a intervenção humana. Através disso, tornou-se possível criar um ambiente de monitoramento capaz de coletar e ceder métricas em painéis atualizados constantemente, fornecendo alta disponibilidade e controle para o usuário final.

Para conquistar esse objetivo, cinco objetivos específicos foram traçados:

- ➢ Fundamentação e estudo de sistemas de monitoramento e suas métricas;
- ➢ Análise e compreensão das ferramentas de monitoramento;
- ➢ Definição de ferramentas e arquitetura do ambiente;
- ➢ Criação de um ambiente de monitoramento;
- ➢ Configuração de recebimento de alertas via Telegram, visando a alta disponibilidade.

1.3 METODOLOGIA

Primeiramente, foi realizado o estudo teórico sobre diferentes tipos de monitoramento disponíveis, capazes de suprirem a demanda de alertas para que fosse possível uma futura implementação do ambiente. Também, foi nessa etapa que métricas importantes em um ambiente de infraestrutura foram apuradas. Dessa forma, tornou-se possível expor a distinção entre plataformas, tanto quanto mostrar os principais recursos que foram trabalhados a fim do desenvolvimento do trabalho.

Na segunda etapa, foi feito um estudo teórico sobre ambientes Linux e seus benefícios. Além disso, nessa etapa foi feita a definição sobre a comunicação entre a plataforma de monitoramento e a plataforma de comunicação capazes de prover alertas em tempo real para o usuário final.

Passando para a terceira etapa, foram feitas as escolhas das ferramentas que compuseram o ambiente de monitoramento, tanto quanto o estudo dos requisitos sobre a criação de máquinas virtuais capazes de prover alertas de infraestrutura gerados por um ambiente Linux. É nessa etapa que ocorreu o planejamento da implementação de uma plataforma de comunicação capaz de receber alertas em tempo real.

Na quarta etapa, foi feito o desenvolvimento do ambiente de monitoramento e suas parametrizações, juntamente com a implementação de códigos capazes de resolver alertas em tempo real sem a necessidade de intervenção humana. Com isso,

foi possível oferecer uma maneira diferente e fortemente escalável a fim de colaborar com a produtividade da aplicação, tanto quanto com a satisfação do usuário.

Nas etapas finais, foram feitos testes e validações de resultados que foram ponderados e homologados. Após a homologação, foi feita a apresentação dos resultados obtidos, reforçando todas as suas funcionalidades e vantagens.

1.4 ESTRUTURA DO TRABALHO

Este trabalho foi dividido em sete capítulos, fora o capítulo que trata a introdução, os quais versam os temas a seguir.

No capítulo 2 do presente trabalho foi realizada a fundamentação teórica dos tipos de monitoramentos de servidores mais utilizados, os diferentes tipos de servidores que podem ser monitorados e os elementos de maior importância que compõem um ambiente de infraestrutura.

No capítulo 3, foi realizado um estudo sobre as plataformas de monitoramento de código aberto e código fechado. Além de expor os tipos de casos, foi apresentado plataformas com diferentes formas de coletas envolvendo agentes, juntamente suas particularidades.

No capítulo 4, foi feita a definição dos parâmetros para a escolha das plataformas de monitoramento, juntamente com a seleção de ferramentas a serem utilizadas neste trabalho.

No capítulo 5, ocorreu o desenvolvimento do ambiente de monitoramento. Foi neste capítulo que foi realizada a preparação, instalação e parametrização de softwares, configurando ferramentas e métricas do monitoramento.

No capítulo 6 foram feitas as validações e casos de testes, de forma que as funcionalidades previstas pudessem ser implementadas, explicando suas utilidades no monitoramento de servidores.

Por fim, no capítulo 7, foram feitas as considerações finais que deram sentido a este trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Antes mesmo de começar a representação de modelos de monitoramento, é importante que seja feita uma descrição do funcionamento das ferramentas, para que seja possível a continuação do trabalho. Portanto, entender os conceitos básicos é extremamente necessário para qualquer ocasião que esse tipo de tecnologia seja utilizado.

Abaixo estão alguns itens essenciais para a compreensão do tema, será uma breve introdução para contextualizar a base do monitoramento, tanto quanto, informações ricas em conteúdo que abrangem diferentes áreas da tecnologia da informação.

2.1 SISTEMAS DE MONITORAMENTO

Atualmente, está cada vez mais necessário o uso de ferramentas de monitoramento na área da tecnologia, que sejam capazes de entregar informações relevantes ao usuário. A principal função do monitoramento de infraestrutura é o acompanhamento de uma aplicação, para sempre propiciar o objetivo de alta disponibilidade e controle. As ferramentas de monitoramento são usadas para fazer coletas constantes de aplicações aptas para a identificação de adversidades antes mesmo que o usuário reconheça (RAKOSHITZ, 2003).

É notável o crescimento interno da utilização de redes de computadores nos dias atuais. Com essa demanda, os obstáculos aumentam na mesma dimensão, como por exemplo: problemas de redes ineficientes, problemas de *backup*, alta utilização de *CPU*, dentre outros (NEVES, 2019). Não via de regra, empresas possuem profissionais aplicados amplamente ao monitoramento. São esses os profissionais encarregados de fazer a solução de adversidades provindas pelo sistema de monitoramento que estejam impactando a disponibilidade empresarial (MOHR, 2012).

Com isso, é válido ressaltar que conforme as demandas de alertas cresçam, é preciso soluções capazes de suprir esse crescimento. Dessa forma, é oportuno saber a importância em competir com as plataformas de código aberto (URLOCKER, 2009), uma vez que o coletivo corrobora com o crescimento, tanto quanto com os seus erros

e falhas. Com essas informações, é válido o incentivo para a busca de soluções de código aberto no mercado, para que seja possível filtrar quais são as mais convenientes de acordo com o propósito de pesquisa.

Além da escolha de utilizar diferentes plataformas para coleta de dados, é importante o entendimento de diferentes tipos de monitoramento, a fim de coletar qual modelo atende a necessidade proposta. Conforme Gaidargi (2019), existem três categorias básicas de monitoramento, relacionado com o ramo empresarial:

- *Monitoramento Técnico;*
- *Monitoramento Funcional;*
- *Monitoramento de Processos de Negócios.*

2.1.1 Monitoramento Técnico

O monitoramento técnico visa a plenitude de equipamentos ou *softwares* próprios. O mesmo está vinculado com a função do objeto, e não na execução do sistema. Sem dúvidas, é o mais importante dos três citados acima, uma vez que além de desvendar o que está acontecendo de falso (anormal) no ambiente, ele tenta consertar o imprevisto. Isso deve-se à projeção de ferramentas de monitoramento criadas com o objetivo de executar funções (GAIDARGI, 2019).

2.1.2 Monitoramento Funcional

O monitoramento funcional investiga a finalidade entregue pelo aplicativo ou pelo sistema distribuído. Com isso, o objetivo do monitoramento funcional é pressupor o desempenho e a disponibilidade da aplicação.

Usualmente, o monitoramento funcional está vinculado à configuração de códigos para andamento em um sistema, o que o torna um excelente instrumento para a geração de relatórios e controle de andamento quando o objetivo está unido com a qualidade do serviço entregue. Em outras palavras, esse monitoramento está submetido à identificação de um problema ou não, conseguindo filtrar a execução e flexibilidade do sistema. Porém, o mesmo não soluciona problemas, é apenas uma forma casual para coletarmos o que está acontecendo e gerando impacto no sistema (GAIDARGI, 2019).

2.1.3 Monitoramento de Processos

O monitoramento de processos de negócios tem grande importância quando o assunto é gerenciamento de sistemas, é nele que grandes empresas oferecem a solução como ferramenta de monitoramento. É cabível o entendimento que a maioria das operações incluem seres humanos, diferentemente do monitoramento funcional, mas da mesma forma tem o objetivo de ser entregue para o cliente. Esse tipo de monitoramento consegue disponibilizar informações de longo prazo para as empresas, juntamente com informações relevantes para entendimento do estado atual de como a empresa está se comportando (GAIDARGI, 2019).

2.2 SERVIDORES

A expressão servidor determina um meio projetado para autuar e conceder dados via outro computador ou em uma rede local (MITCHELL, 2021). É nele que é definido uma maneira que possibilita o processamento de informações e armazenamento de dados. O termo servidor está relacionado a uma máquina física, uma máquina virtual, ou um *software* que está performando serviços. (POSEY, 2021) Os sistemas ainda têm a opção de serem físicos ou virtuais, com alto processamento de dados. Conforme Posey (2021), é possível ter um melhor entendimento sobre tipos comuns de servidores:

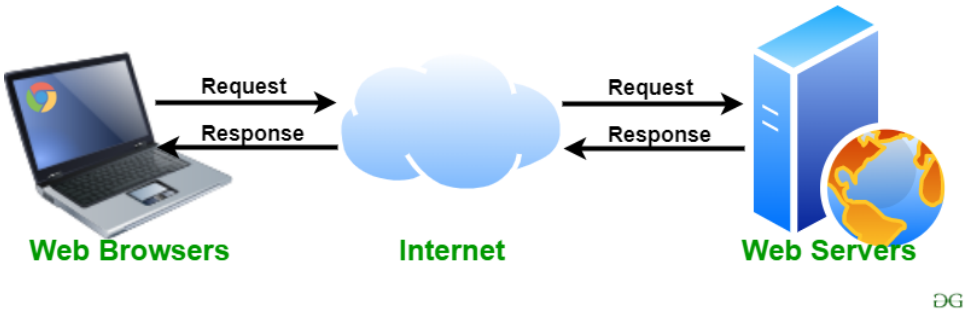
- *Servidor web;*
- *Servidor de email;*
- *Servidor FTP;*
- *Servidor de banco de dados.*

2.2.1 Servidor Web

Um servidor *web* (Figura 3) é capaz de apresentar páginas e aplicativos através de navegadores *web* via protocolo *http*. Grande parte dos servidores que estamos acostumados a nos conectar diariamente via navegador estão ligados a um

web server, tais como Facebook, Twitter, sites de notícias, entre outros. Além de serem utilizados para diversas tarefas simples, tais como escrita e texto e *upload* de imagens, vale ressaltar que tarefas mais complexas como por exemplo serviços de *backup online* (*backup cloud*) são suportados também.

Figura 3 - Web server request and response



Fonte: https://www.geeksforgeeks.org/web-server-and-its-type/ Acesso em: 11 maio 2021

2.2.2 Servidor de Email

O servidor de email, como pode ser visto na Figura 4, está ligado ao envio e recebimento de mensagens de email em uma rede. Grande parte dos *clients* de emails se conectam via protocolos *imap* ou *pop3* para *download* de mensagens localmente e via protocolo *smtp* para o envio de mensagens através de um servidor de email (HOLTZ, 2021).

Figura 4 - Client and email server

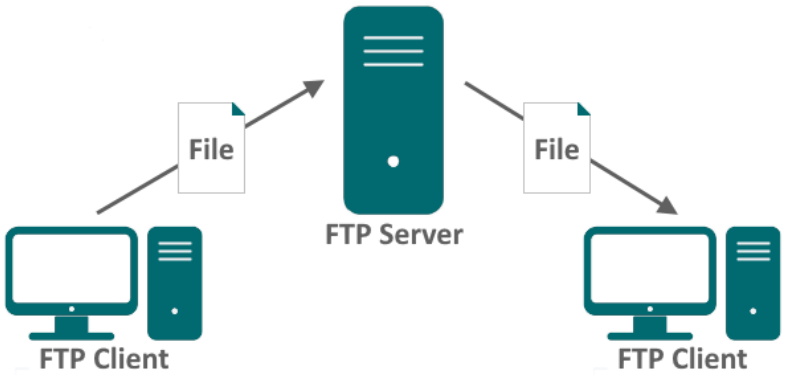
Fonte: https://www.samlogic.net/articles/mail-server.htm Acesso em: 10 maio 2021

2.2.3 Servidor FTP

Os servidores FTP, relatado na Figura 5, deslocam arquivos através das próprias ferramentas do protocolo FTP. É possível acessá-los remotamente através de programas clientes de FTP que sejam possibilitados à conexão de

compartilhamento de arquivos. Esses servidores são capazes de executar duas tarefas básicas chamadas de *put* e *get* resultando em uma solução boa e barata para fazer a transferência de arquivos.

Figura 5 - FTP server and clients



Fonte: https://ozekirobot.com/p_6047-ozeki-robot-status-updates-on-a-website-using-ftp.html Acesso em: 11 maio 2021

2.2.4 Servidor de Banco de Dados

Os servidores de banco de dados são essenciais para o fluxo e preservação dos dados que estão relacionado com aplicativos e programas. É neles que se encontram informações de dados e gerenciamentos de usuários e dispositivos. Através dessa importância, o termo segurança é altamente explorado, para que seja possível ter o controle referente riscos e integridade (INGALLS, 2021).

Através desse tipo de servidor é possível lidar com grandes quantidades de dados regularmente, uma vez que a arquitetura cliente-servidor é frequentemente preenchida por processamento de dados. O DBMS (*Database Management System*) fornece acesso ao banco de dados pelo modelo cliente-servidor, tornando o acesso possível para clientes acessarem o servidor através de uma aplicação *frontend*, que irá exibir o conteúdo relacionado, ou também é possível através de um *backend* que rode e administre o servidor.

Figura 6 – Database Server

Fonte: https://3.imimg.com/data3/LQ/GC/MY-8728205/database-server-500x500.jpg
Acesso em: 11 maio 2021

2.3 MÉTRICAS DE SERVIDORES MONITORADOS

As métricas de monitoramento são as medidas determinadas em um sistema de monitoramento capazes de coletar dados referentes a serviços ou de uma infraestrutura a fim de administrar impactos indesejados no comportamento de alguma alteração feita (Ellingwood, 2017).

Para que isso seja possível, as métricas mais utilizadas por provedores de infraestrutura são a utilização de *CPU*, utilização de memória e informações referentes a processos internos (NAYAK, 2020). Além de todas as métricas que os sistemas de monitoramento são capazes de reconhecer, é possível fazer a captura de eventos específicos, ajudando o profissional de computação a coletar informações não fixas, tanto quanto, ter um histórico de todos os acontecimentos gerados em um servidor, tornando assim um excelente meio de análise para prevenção de eventos indesejados.

Ao coletarmos métricas, podemos classificá-las em dois grupos, de baixo nível, que geralmente são oriundas do próprio sistema operacional, e métricas de alto nível, que tem uma maior relação com a parte de aplicações, consequência de execuções de um determinado sistema (Sematext, 2020).

Além de ter em mente o que precisa ser monitorado, é preciso saber quais as métricas que se deve coletar, deixando claro ao usuário para facilitar a interpretação. Outro aspecto importante, é a frequência de coleta de dados implementada em um

sistema de monitoramento, pois dependendo da criticidade do alerta, pode ser ocultado por um tempo prolongado, isso porque a granularidade intensa é um fator que pode ocasionar uma carga indevida no desempenho da aplicação, tanto quanto no armazenamento de dados.

2.3.1 CPU

O consumo de *CPU* é um relato de intensidade de como os processadores estão sendo utilizados. Essa utilização pode variar de acordo com a quantidade de tarefas que estão sendo executadas em determinados momentos por um sistema operacional ou um programa. Porém, a alta utilização de CPU nem sempre é significado de problemas, desde que as razões estejam sob conhecimento e que não afetem a aplicação no longo prazo, mas sempre que possível, é recomendada a tentativa de diminuição de uso alto de CPU (Ionos, 2020).

Conforme Haynes (2019) e Kreeftmeijer (2018), além do monitoramento de métricas principais de CPU ser dividido entre *system* e *user*, existem subcategorias que precisam ser vistas para a melhor coleta e entendimento de dados de um sistema. Abaixo estão as subcategorias mais relevantes, segundo os autores:

- *System:* Apresenta a quantidade de tempo utilizada pelo *kernel* juntamente com as tarefas do *kernel mode*, podendo variar de acordo com a quantidade de entrada e saída de dados que estão sendo gravados no disco.
- *User:* Expõe a quantidade de tarefas que estão sendo executadas pelo *user-mode* contendo o código de aplicação. Porém, se a aplicação pretende ler ou escrever algum arquivo na rede, ele entrará em um modo de intervalo até que o *kernel* finalize a execução até o momento da volta.
- *Nice:* Refere-se à quando a *CPU* estiver executando alguma tarefa do usuário com uma prioridade abaixo do padrão, priorizando as tarefas mais importantes.
- *Idle:* Ocorre quando não há nenhuma atividade para o *kernel*, tendo como resultado nenhuma operação de entrada e saída de dados.

- ➢ *I/O Wait:* Acontece quando a *CPU* só tem uma tarefa de espera de entrada e saída de dados a ser feita.
- ➢ *Steal:* Ocorre principalmente em ambientes virtualizados quando o *hypervisor* (processo que cria máquinas virtuais) consome ciclos de *CPU*, dando para outros inquilinos por diferentes razões e motivos.
- ➢ *IRQ e SoftIRQ:* É resultado de quando o *kernel* está obedecendo a solicitações de interrupções.
- ➢ *Guest e Guest Nice:* É quando o *hypervisor* está executando uma *CPU* virtual (já incluindo na subcategoria *user* e *nice*).

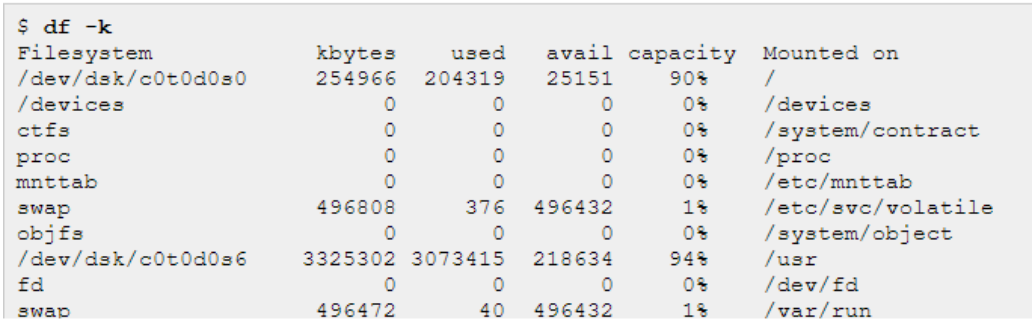
2.3.2 Espaço em disco

A capacidade de disco ou espaço em disco, é a capacidade total de dados que o disco consegue armazenar. É uma métrica bastante utilizada em ambientes de infraestrutura, normalmente exibida em *megabytes*, *gigabytes* ou *terabytes* (Hope, 2020). Além disso, termos como espaço total (quantidade total da partição) e espaço ocupado (quantidade consumida) são frequentemente utilizados.

Em ambientes Linux, é comum se ver o ambiente separado em particionamentos, tendo como principais características a permanência do espaço total do disco, mas com diferentes tipos e capacidades em diferentes partições. A Figura 7 demonstra, através do comando "*df -k*", os diferentes particionamentos em um ambiente Oracle Linux, sendo uma opção possível para customização com o objetivo de melhorar o desempenho de um sistema. Abaixo estão os cinco tópicos para entendimento da Figura 7:

- ➢ *Kbytes:* Espaço total que pode ser utilizado no sistema.
- ➢ *Used*: Quantidade de espaço utilizado.
- ➢ *Avail:* Quantidade de espaço restante para ser utilizado.
- ➢ *Capacity*: Quantidade em percentual da utilização total do disco.
- ➢ *Mounted on:* Ponto de montagem em que a partição se encontra.

Figura 7 - df -k command



```
$ df -k
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c0t0d0s0     254966  204319   25151    90%    /
/devices                   0       0       0     0%    /devices
ctfs                       0       0       0     0%    /system/contract
proc                       0       0       0     0%    /proc
mnttab                     0       0       0     0%    /etc/mnttab
swap                  496808     376  496432     1%    /etc/svc/volatile
objfs                      0       0       0     0%    /system/object
/dev/dsk/c0t0d0s6    3325302 3073415  218634    94%    /usr
fd                         0       0       0     0%    /dev/fd
swap                  496472      40  496432     1%    /var/run
```

Fonte: https://docs.oracle.com/cd/E18752_01/html/817-0403/spmonitor-6.html Acesso em: 15 maio 2021

2.3.3 Utilização de memória

O primeiro modelo de memória mais conhecido em ambientes de infraestrutura é a memória RAM, usado principalmente para armazenamento de arquivos e programas durante o processo de utilização do servidor. Grande parte da sua importância está no momento de leitura e gravação, pois quando comparado com outros tipos de armazenamento, como por exemplo HD ou SSD, a memória RAM consegue ter uma grande vantagem (Peterson, 2020).

Outro modelo bastante conhecido é a *swap*, cuja função fundamental é substituir o disco quando não há mais espaço disponível na memória RAM, fazendo assim uma troca, liberando memória física para a alocação (Both, 2020). Essa responsabilidade de troca entre as memórias é de autoria do *kernel*, porém a desvantagem é que além da vida útil do disco ser comprometida, os discos são extremamente mais lentos quando comparados à memória RAM (Sims, 2007).

Para isso, existem diferentes métricas que podem ser verificadas e adicionadas em um sistema de monitoramento para coleta de dados referentes à memória (Linuxize, 2020):

- *Total:* Capacidade total de memória que pode ser utilizada.
- *Used*: Memória utilizada, composta por *used* = *total* - *free* - *buffers* - *cache*.
- *Free:* Quantidade de memória livre.
- *Shared*: Compatibilidade de versões prévias.

- ➢ *Buff/cache:* Capacidade total de memória que pode ser utilizada por *Buffers* e *Cache*.
- ➢ *Available:* Suposição de memória disponível para novas aplicações sem utilizar *swap*.

Já as métricas que podem ser utilizadas no monitoramento *swap* são:

- ➢ *Free:* Capacidade desocupada da memória virtual.
- ➢ *Used*: Capacidade utilizada da memória virtual.

Conforme visto acima, a parte de gerenciamento de memória é um tópico essencial em um ambiente de infraestrutura. Com uma organização adequada, é possível monitorar diferentes métricas distintas no que diz respeito à memória.

2.3.4 Monitoramento de Rede

A prática de monitoramento de redes de infraestrutura proporciona às empresas soluções de automatização e verificação de integridade para que a resolução de alertas seja feita de maneira ágil e competente. Com isso, coloca-se à disposição do usuário ferramentas de *software* capazes de fazerem o processo de coleta de informações, tais como tráfego, atualização de status e consumo de banda da internet (Cisco, 2021).

Decidir quais os itens monitorar em uma rede é uma ótima forma para evitar situações inesperadas. Através dessa decisão, servidores, aplicações e *desktops* desempenham um futuro mais simples quando se há clareza no mapeamento do monitoramento. Alguns sistemas de monitoramento tem a função de descoberta automática, tendo relação com a capacidade de registro de dispositivos (endereço IP, serviços, roteadores) à medida que são adicionados, removidos ou que são submetidos a alguma alteração de configuração (NASH, 2007).

Segundo Shamsi (2009), existem duas abordagens básicas de monitoramento de rede:

- *Monitoramento ativo*
- *Monitoramento passivo*

O monitoramento ativo também pode ser chamado de monitoramento sintético. É nele que é simulado o comportamento do usuário para determinar o possível desempenho da rede. No monitoramento ativo, existe o controle total sobre o intervalo de monitoramento, tamanho dos pacotes e os caminhos a serem monitorados, ou seja, o monitoramento ativo cumpre a função de analisar os dados de desempenhos simulados.

A diferença entre o monitoramento ativo e passivo é que o passivo coleta dados reais dos usuários para que seja analisando um momento específico. Dessa forma, é possível que o monitoramento passivo possa coletar grande quantidade de dados, pois ele não é executado com tanta frequência, quando comparado com o monitoramento ativo. A preferência pelo tipo de monitoramento vai variar de acordo com a aplicação e a abordagem que a mesma tenha que ser analisada.

2.4 AMBIENTES LINUX

Os servidores Linux estão se tornando cada vez mais difundidos dentro das empresas. Segundo Michel Fischer, CEO da Metisentry, o Linux estava ambientado a ser um sistema operacional caracterizado pelos seus detalhes, e o seu uso está fortemente implantado no ambiente de desenvolvimento, ambientes corporativos e principalmente em servidores web. Porém, nos últimos anos as plataformas de código aberto, incluindo o Linux, tornaram-se uma excelente opção quando o assunto está ligado à substituição de servidores Microsoft, quebrando um pouco do paradigma de predominância da empresa referentes aos ambientes comerciais. À medida que os *data centers* se ampliam, a parte de custos referentes aos licenciamentos da Microsoft aumentam na mesma proporção (Burroughs ,2017).

Abaixo é possível ter um melhor entendimento dos benefícios que uma distribuição Linux pode trazer para o mercado de trabalho, juntamente com o crescimento escalável que está sendo utilizado por centenas de milhares de pessoas em todo o planeta (Sathyanarayanan, 2020).

O primeiro e mais importante é o custo. Se compararmos os produtos da Microsoft em questão de valores com as distribuições Linux, é possível vermos que os produtos Microsoft estão à disposição por taxas altas. Na maioria das vezes, as licenças são instaladas somente em um computador, enquanto o preço de instalação de uma distribuição Linux é custo zero (Kumar, 2015), tornando um excelente dado para as empresas. Outro aspecto importante é que os softwares básicos que são necessários para um usuário padrão já estão disponíveis, com isso competências e demandas que necessitem edição de fotos, edição de áudio e edição de vídeo estão disponíveis ao usuário final (Sathyanarayanan, 2020).

O segundo benefício está relacionado à segurança. Nenhum sistema operacional está totalmente protegido, isso porque a proteção está diretamente associada à forma que o usamos (Mudrakola, 2021). No sistema operacional Linux desde o início dos anos 90, foi possível se manter seguro contra espalhamento de vírus, *spyware* e *adware* (Kumar, 2015). Um fator que colabora fortemente com essa segurança é a estrutura do sistema operacional ser código aberto, isso porque a comunidade Linux está constantemente mantendo melhorias nos pacotes essenciais para o funcionamento do sistema, sendo reflexo de segurança e precaução para quem estiver utilizando.

O terceiro, mas não menos importante benefício, é o poder de escolha que o usuário tem sobre diversas distribuições Linux existentes. A escolha faz-se possível de acordo com a proposta no momento da instalação. Termos como familiaridade, simplicidade e requisitos do servidor são levados em consideração para que a escolha seja certeira. Embora existam distribuições como a Red Hat, o OpenSuse é extremamente diferente de todas as outras distribuições Linux, isso deve-se ao fato de o mesmo depender de ferramentas de configurações que já foram prejudicadas com o tempo, tais como YaST e também sobre o envolvimento da Novell com o SUSE Linux (Venezia, 2014), que virou uma distribuição pouco utilizada nos dias de hoje em consequência de suas particularidades de incompatibilidade com diversos pacotes essenciais.

2.5 PONDERAÇÕES SOBRE O CAPÍTULO

Com base no conteúdo do capítulo, pode-se dizer que o monitoramento de servidores é extremamente relevante para um ambiente empresarial. Através de um

ambiente monitorado, obtém-se facilidade na coleta de informações relevantes para o funcionamento da estrutura, colaborando com a alta disponibilidade proveniente de parametrizações corretas feitas pelo administrador.

Através do grande crescimento na demanda visando a saúde do ambiente monitorado, diferentes tipos de monitoramento foram definidos e consequentemente mais plataformas foram sendo desenvolvidas. Tais plataformas foram criadas com o intuito de coleta e envio, para que atendam as métricas definidas através do planejamento da arquitetura do ambiente.

Pode-se monitorar o servidor através das métricas de baixo e alto nível. As métricas de baixo nível estão relacionadas as informações coletadas do sistema operacional, tais como: utilização de CPU, espaço total em disco, memória livre e a saúde do tráfego de rede. As métricas de alto nível tem relação com as aplicações que estão sendo executadas através do servidor, sendo geralmente processos mais customizados pelo dono da aplicação.

O estudo deste capítulo de fundamentação teórica colaborou com a compreensão das diferentes métricas que os servidores são capazes de proporcionar, coleta e análise de dados, sendo a primeira etapa para o atingimento do objetivo deste trabalho.

3 FERRAMENTAS DE MONITORAMENTO

Logo após o aprendizado sobre a fundamentação teórica, é de se entender a efetividade de uma boa configuração de alertas para as empresas de tecnologia. As ferramentas de monitoramento do sistema Linux têm a função de assegurar que o ambiente esteja normal, ajudar a identificar condutas estranhas e desempenhos anormais da aplicação (Nichols, 2021).

Com a finalidade de atingir um controle ideal na hora da execução, é de extrema importância que os administradores do ambiente sejam capazes de controlar da forma mais prática possível. Grande parte dos monitoramentos de sistemas incluem a instalação de um *software* de gerenciamento no dispositivo que precisa ser monitorado e autenticado em um servidor de monitoramento (Veesp, 2017), fazendo com que seja possível a parte da configuração. É através de um *software* de monitoramento que se encontram os mecanismos necessários para se fazer a varredura e coleta de numerosos dados de sistemas ao longo de um período determinado de tempo determinado pelo administrador do ambiente.

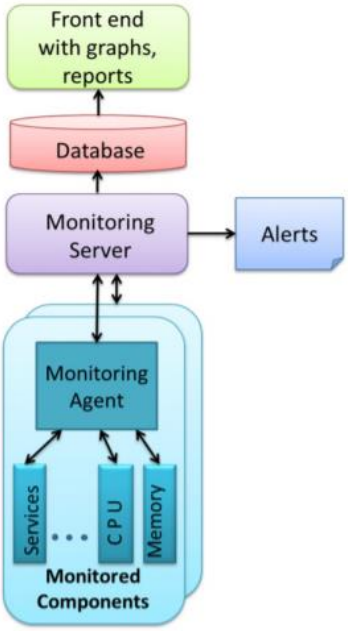
Segundo Campion (2021), existem dois tipos de ferramentas que são possíveis de serem utilizadas para o monitoramento:

- ➢ Ferramentas de monitoramento de código aberto;
- ➢ Ferramentas de monitoramento proprietária

3.1 CARACTERÍSTICAS DAS FERRAMENTAS

Uma das principais características do monitoramento de uma infraestrutura é ter uma arquitetura bem definida, dessa forma, a compatibilidade com ambientes Linux, Windows, entre outros é de suma importância para que os alertas em tempo real consigam ser transmitidos para uma tela final (*frontend*) e o administrador de rede consiga controlar através da interface.

Figura 8 - Infrastructure monitoring tool architecture



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

Na Figura 8, segundo Fatema (2014, p.2922), é possível identificar que ferramentas de monitoramento, em grande parte, utilizam um modelo gerente-agente característico. A arquitetura possui um banco de dados, responsável pelo armazenamento das informações coletadas, um painel de *frontend* responsável por exibir informações, gráficos e alertas que impactam diretamente em um servidor que esteja com o monitoramento configurado.

Segundo Hein (2020), existem sete razões essenciais para se escolher soluções de monitoramento:

- *Métricas em tempo real;*
- *Descoberta automática de dispositivos;*
- *Alertas inteligentes;*
- *Diagnósticos de erros e análises;*
- *Mapas de rede;*
- *Painéis customizáveis;*
- *Escalabilidade*

Sobre a velocidade, é possível também ter acesso a diversos plugins, que são ferramentas que permitem resolver problemas rapidamente conforme necessário, além de acrescentar flexibilidade à aplicação (STERNE, 2015). Caso o administrador da aplicação não tenha experiência técnica, é possível o contato com uma comunidade ativa, tanto quanto, participar e obter sugestões para o auxílio rápido da resolução do problema.

No termo qualidade, o código aberto atende o usuário final pelo principal motivo que é a comunidade que conduz a mudança e melhoria, e isso permite à comunidade criar o que bem entender. Por ter um domínio maior, é possível ter produtos de maior qualidade, e é por causa dessa superioridade tecnológica que as principais empresas do mundo (Google, Facebook e AWS) utilizam ferramentas de código aberto (King, 2021).

Ao descontinuar o uso do *software* de código-fonte fechado, muitas empresas precisam encontrar uma alternativa (caso haja) para que o monitoramento possa continuar operando. Essa descontinuação pode acontecer em qualquer *software*, mas em código aberto, pode ser acessado e o projeto pode ser continuado. Se caso as empresas proprietárias de *softwares* de monitoramento interromperem o investimento, então, se houver vantagens, há de se ter uma comunidade ativa para dar continuidade ao projeto.

A maior parte das empresas não percebe que, ao considerar alternativas de código aberto, podem obter *softwares* e serviços de maior qualidade gratuitamente ou a um custo razoável com o investimento orçado. Os *softwares* são geralmente fornecidos gratuitamente, mas também há a opção de doação para a equipe de desenvolvimento ou para apoiar as assinaturas relacionadas com o produto. Em comparação com os *softwares* comerciais, os produtos de código aberto pagos existentes geralmente são vendidos a um preço alto (King, 2021).

Os gastos com produtos comerciais de código-fonte fechado são altamente fundamentados, e com isso as empresas proporcionam essa fundamentação de informações para fornecer uma sensação de falsa segurança. Com isso, os desenvolvedores consistem fortemente na utilização de *softwares* de código aberto, e as empresas estão satisfeitas com os resultados, porque projetos em código aberto conseguem ter um maior impacto na comunidade (Korolov, 2018).

As ferramentas proprietárias de monitoramento são licenciadas, com isso, o custo para utilização pode ser extremamente alto. Uma vantagem muito potente de

ferramentas licenciadas é que as configurações de instalações são muito práticas, portanto, seus recursos de monitoramento podem ser aumentados ou diminuídos com base no tamanho do negócio ou em outros requisitos envolvendo a aplicação da empresa (Melnick, 2021).

3.2 COLETA DE DADOS

As soluções de monitoramento geralmente podem ser comprovadas por detecção precoce de erros, mal funcionamento e alertas frequentes, para que uma intervenção imediata possa ser feita. Além disso, os administradores da rede não precisam mais monitorar continuamente todos os componentes de uma estrutura para garantir a alta disponibilidade.

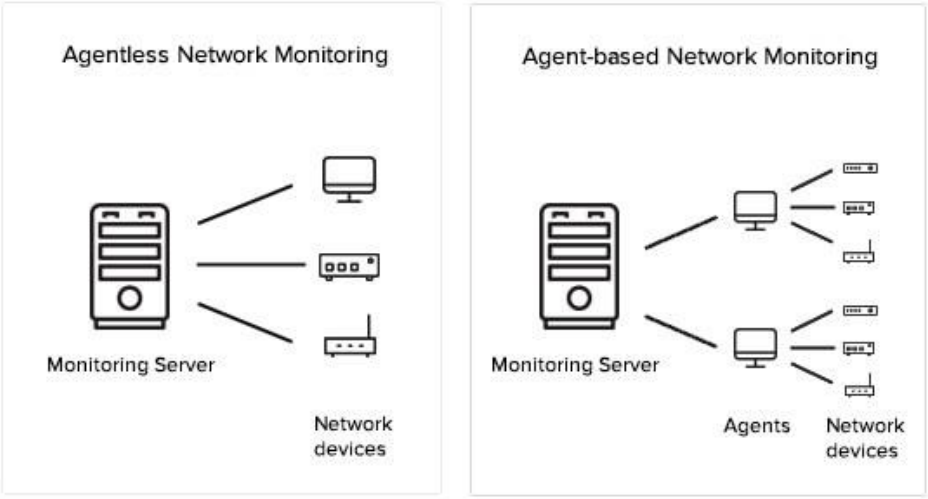
3.2.1 Agentes

Segundo Suvvari (2018), são duas as principais categorias nas quais o usuário é capaz de escolher:

- *Monitoramento sem agente;*
- *Monitoramento com agente;*

O monitoramento sem agente pode ser implementado de duas formas diferentes, usando uma API exposta pela plataforma ou serviço que está sendo monitorado, ou analisando diretamente os pacotes de rede que estejam entre os componentes do serviço. Contudo, a análise de rede não irá fornecer detalhes de métricas sobre os servidores que oferecem suporte aos serviços de aplicativo que se comunicam pela rede, mas fornecerá dados sobre o desempenho e a disponibilidade do serviço.

Figura 9 - Agentless Monitoring and Agent-based Monitoring



Fonte: https://www.manageengine.com/network-monitoring/agentless-network-monitoring.html Acesso em: 15 nov 2021.

Na Figura 9, é possível ver que o administrador da rede pode escolher quais partes da infraestrutura serão monitoradas sem o agente.

O monitoramento com agente permite a coleta mais profunda e ampla de dados, consistindo em componentes de *software* (geralmente pequenos aplicativos) que residem no *client server* e coletam dados. Em seguida, os dados são devolvidos para a estação de monitoramento de acordo com a estratégia do administrador, mas abordagens como essa resultam em indicadores lapidados de monitoramento, armazenados em banco de dados, alertas e relatórios, bem como análises mais profundas para uma possível correção de problemas.

Com isso, é possível fornecer grande parte das soluções como *SaaS,* tendo maior facilidade de acesso aos *softwares* via *web*. Os agentes que estiverem configurados na aplicação vão coletar individualmente cada dado e retornar para o *proxy*. Dessa forma, a vantagem é que, em comparação com o uso de um sistema sem agentes, os dados coletados são extremamente mais detalhados, porque o agente geralmente tem um nível mais alto de acesso ao *hardware* configurado. Porém, a desvantagem é que a instalação do agente precisa ser feita de maneira individual em cada dispositivo que queira ser monitorado, necessitando de compatibilidade correta, tais como versões, sistemas operacionais, *softwares* entre outros.

Figura 10 - A Zabbix agent running under Linux:

```
# ps u -C zabbix_agentd
USER       PID %CPU %MEM    VSZ   RSS   STAT TIME COMMAND
zabbix   15778  0.0  0.0  48212   460   SN   0:00 /usr/sbin/zabbix_agentd
zabbix   15780  0.0  0.0  48212   748   SN   9:27 /usr/sbin/zabbix_agentd
zabbix   15781  0.0  0.0  48212   424   SN   0:00 /usr/sbin/zabbix_agentd
zabbix   15782  0.0  0.0  48212   424   SN   0:00 /usr/sbin/zabbix_agentd
zabbix   15783  0.0  0.0  48212   424   SN   0:00 /usr/sbin/zabbix_agentd
zabbix   15784  0.0  0.0  48220   612   SN   0:17 /usr/sbin/zabbix_agentd
```

Fonte: https://www.zabbix.com/zabbix_agent Acesso em: 22 maio 2021

Através da Figura 10, podemos ver que o agente do serviço de monitoramento Zabbix está instalado em um servidor Linux, coletando dados em tempo real e enviando para um *proxy*, na qual está ligado a uma interface *web* que pode ser visualizada por administradores do monitoramento sempre que for necessário.

3.3 PONDERAÇÕES SOBRE O CAPÍTULO

As ferramentas de código aberto são excelentes escolhas de monitoramento devido aos seus benefícios, principalmente quando os objetivos de escolha estiverem associados a orçamento restrito e colaboração da comunidade através de fóruns e trocas de conhecimento.

Dependendo do caso, as ferramentas de código privado podem ser uma boa escolha também, mas é válido lembrar que caso a empresa fornecedora do serviço de monitoramento descontinue as versões, pode ser um problema a longo prazo para uma empresa que já possui toda a arquitetura de monitoramento formada.

Com relação a coleta de dados, é possível optar por plataformas com ou sem agentes. Porém, os casos de utilização de plataformas com agentes são maioria, devido a coleta mais profunda e ampla dos dados através da instalação no servidor.

A grande quantidade de referências deste capítulo proporcionou pontos e observações distintas, mostrando que dependendo do propósito de construção de monitoramento, pode-se optar por formas diferentes de coleta, mas sempre visando informações vindas do ambiente monitorado.

4 SELEÇÃO DAS FERRAMENTAS

Com base nos capítulos vistos anteriormente sobre os componentes de gerenciamento de redes, foi preciso estabelecer critérios para que seja possível o desenvolvimento do trabalho proposto. Dessa forma, é válido ressaltar que métricas como coleta de dados, armazenamento de dados, visualização de alertas e resolução de problemas precisaram ser tratadas com prioridade, e pode ser realizada através de três tipos específicos de ferramentas:

- ➢ Ferramentas coletoras: Trata-se de *softwares* singulares capazes de coletar métricas importantes nas quais precisam ser monitoradas.
- ➢ Ferramentas de armazenamento: Banco de dados que têm a principal função de armazenar dados, item essencial para as ferramentas de visualização.
- ➢ Ferramentas de visualização: Refere-se a *softwares* específicos capazes de proporcionarem gráficos e alertas para o usuário em tempo real.

Com o propósito de concluir este trabalho, foi feita uma sondagem de ferramentas capazes de proporcionar os três itens citados. Os critérios de escolha foram explicados e mostrados para que pudesse ser aplicado no cenário proposto.

4.1 PARÂMETROS PARA A ESCOLHA DA FERRAMENTA COLETORA

Conforme o interessa da proposta, citado nos capítulos anteriores, a ferramenta coletora deve estar associada a todos os critérios citados abaixo:

I. Compatibilidade com distribuições Linux, necessitando que o *software* de coleta tenha uma versão adaptável ao sistema operacional;
II. Gratuidade: A ferramenta de armazenamento não pode ter custos;
III. Licença open-source: Deve possuir licenças GPLv2, Apache License 2.0, BSD ou MIT;
IV. Estrutura de métricas: O *software* deve permitir a coleta das principais métricas citadas no capítulo 2 (utilização de CPU, uso de memória, espaço em disco e consumo da rede) através de plugins;

V. Capacidade de definir o intervalo de tempo de coleta de dados através de configurações variáveis;
VI. Ferramenta que está sendo continuamente evoluída com a liberação de novos pacotes e atualizações, tornando assim recursos não obsoletos e melhorados.

4.1.2 Parâmetros para a escolha da ferramenta de armazenamento

Segundo o interesse da proposta, a ferramenta de armazenamento, citado nos capítulos anteriores, está coerente a todos os critérios mencionados abaixo:

I. Compatibilidade com distribuições Linux, necessitando que o *software* de coleta tenha uma versão adaptável ao sistema operacional;
II. Gratuidade: A ferramenta de armazenamento não pode ter custos;
III. Licença open-source: Deve possuir licenças GPLv2, Apache License 2.0, BSD ou MIT;
IV. Rotatividade customizada: Possibilidade em ter retenção de dados, de sendo possível limitar o tempo em que dados ficam armazenados;

4.1.3 Parâmetros para a escolha da ferramenta de visualização

Em concordância com o interesse da proposta, a ferramenta de visualização deve suportar os critérios abaixo:

I. Compatibilidade com distribuições Linux, necessitando que o *software* de coleta tenha uma versão adaptável ao sistema operacional;
II. Gratuidade: A ferramenta de armazenamento não pode ter custos;
III. Licença *open-source*: Deve possuir licenças GPLv2, Apache License 2.0, BSD ou MIT;
IV. Interface WEB: Permitir a utilização através de navegadores, em qualquer plataforma;
V. Capacidade de enviar os alertas para um canal de comunicação anteriormente definido;

VI. Compatibilidade com a ferramenta de coleta de dados, para que não tenha impactos negativos na hora da visualização dos alertas

4.2 ESCOLHA DA FERRAMENTA COLETORA

A busca pela ferramenta coletora foi feita em sites como o Compari Tech e o MetricFire. Além de analisar as ferramentas com as suas particularidades, há comparações que possibilitam o estudo e entendimento das plataformas. Dessa forma, as métricas essenciais referentes a monitoramento de infraestrutura foram abrangidas e individualmente discutidas, possibilitando que o usuário entenda as diferenças e opte conforme o seu objetivo.

Para a escolha da ferramenta adequada, as duas ferramentas que se moldaram nos parâmetros essenciais com melhor avaliação foram analisadas e comparadas, tendo como resultado a ferramenta Nagios e o Zabbix.

4.2.1 Nagios

O Nagios é uma ferramenta de monitoramento de código aberto que teve sua breve criação em 1996, por Ethan Galstad. A instalação demanda conhecimentos de códigos-fonte e instalação manual de dependências, além da edição de *templates* de configuração, para que seja possível ter o retorno de todo o gerenciamento estipulado. Desde então, muitas empresas adotaram o uso da ferramenta para monitoramento de infraestrutura por sua usabilidade e comunidade frequentemente ativa.

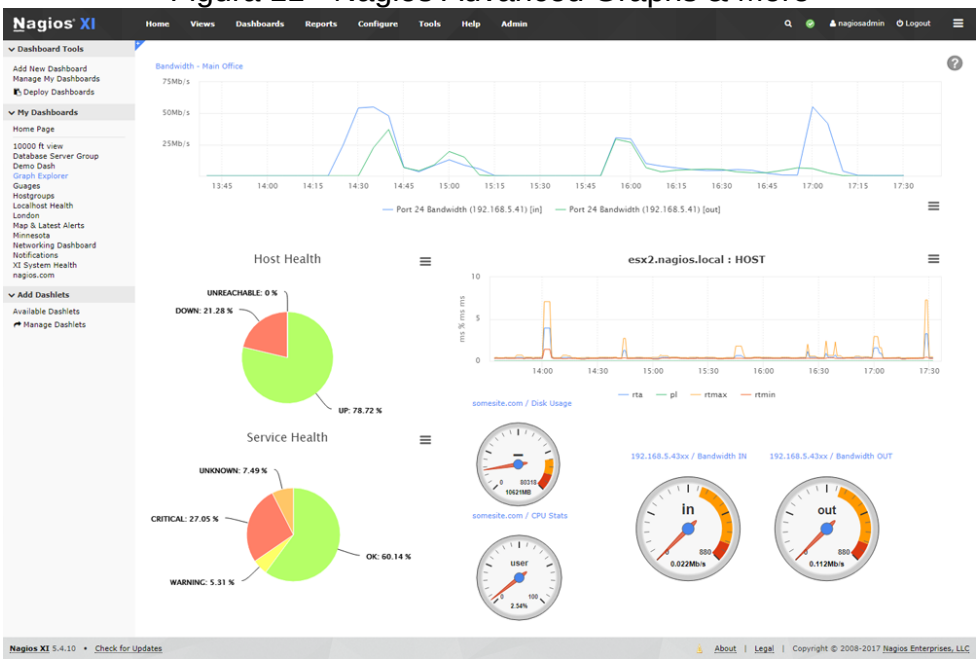
Segundo os documentos (Nagios, 2021) de apresentação da ferramenta, as principais características do Nagios são:

A. Monitoramento completo da infraestrutura;
B. Ambiente customizável através de serviços;
C. Garantir que problemas de infraestrutura tenham efeitos mínimos para a organização.

Todas as características são resultadas de uma arquitetura extremamente organizada, consequência de um sistema elaborado que foi projetado para oferecer confiabilidade e segurança ao usuário final.

A Figura 12 ilustra a *interface web* do Nagios para visualização de métricas e apuração de alertas frequentes que estejam sendo exibidos no momento da coleta.

Figura 11 - Nagios Advanced Graphs & More



Fonte: https://www.nagios.com/products/nagios-xi/ Acesso em: 10 nov 2021

A instalação é descomplicada, na maioria das vezes feita através de pacotes proventos do sistema operacional. São necessários conhecimentos relacionados à transferência de dados dentro de um ambiente Linux, para que ao final da instalação seja possível a parte de configuração e modelagem da aplicação.

4.2.2 Zabbix

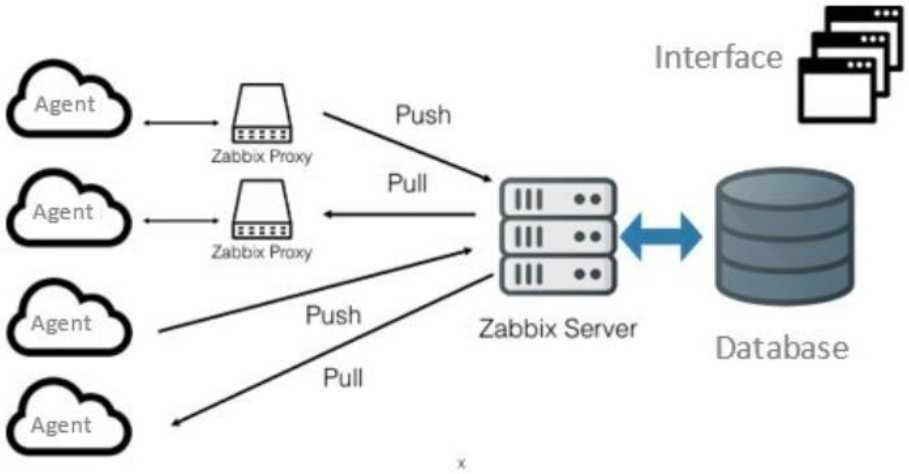
Criado por Vladishev, o Zabbix é constantemente melhorado e ativamente presente em servidores Linux. Além de monitorar diversos parâmetros de rede e infraestrutura como um geral, existe um mecanismo de notificação que possibilita aos usuários configurarem alertas em tempo real e rápidas ações de correção.

O Zabbix Proxy é uma ferramenta capaz de coletar dados de serviços monitorados e enviar esses materiais para o Servidor Zabbix. As informações

coletadas são guardadas localmente, e logo após são enviadas para o servidor Zabbix onde o proxy está instalado.

A Figura 13 descreve o ambiente de arquitetura do Zabbix, sendo um processo totalmente customizável de acordo com os princípios e aplicações do usuário.

Figura 12 - Zabbix Architecture



Fonte: https://postgrespro.com/blog/pgsql/5967895 Acesso em: 28 maio 2021

Segundo a visão geral do manual de arquitetura do Zabbix (2021), a estrutura consiste em seis componentes principais:

I. Zabbix server: É a peça fundamental para que os agentes consigam relatar dados de disponibilidade e estatísticas referentes à aplicação. É nele que as informações operacionais são armazenadas;
II. Database: Todos os elementos de configuração são armazenados em um banco de dados;
III. Interface *web*: É nela que o acesso por meio de gráficos e estatísticas pode ser acessado visualmente, facilitando a interpretação;
IV. Zabbix proxy: Responsável pela coleta de dados, embora opcional em uma arquitetura Zabbix, é extremamente importante para um ambiente com muita demanda de dados;
V. Zabbix agent: Monitoram de forma dinâmica recursos e aplicativos de um ambiente, para que seja possível enviar para o Zabbix Server.

VI. Fluxo de dados: Item essencial que está relacionado à criação de hosts, capazes de se comunicarem e proporcionarem um ambiente flexível para o usuário.

Partindo para a etapa de instalação, o processo é mais complexo de quando comparado ao Nagios, devido aos requerimentos (formas de instalação e compatibilidades) e também pela necessidade de utilizar um banco de dados externo para fazer o armazenamento dos dados.

4.3 ESCOLHA DA FERRAMENTA DE ARMAZENAMENTO

A procura pela ferramenta ideal de armazenamento foi feita através do site DB Engines. Grande parte de informações particulares das ferramentas abordadas são comparadas e analisadas individualmente. Com isso, o modelo de cenário de um monitoramento gratuito fica mais acessível, viabilizando a escolha.

Para a preferência da ferramenta adequada, os resultados foram o MariaDB e o Oracle Database. Duas excelentes ferramentas, mas com diferentes tipos de particularidades e possibilidades de configurações a serem trabalhadas.

4.3.1 MariaDB

Maria DB é um dos servidores de banco de dados mais populares do mundo. Teve sua origem através dos desenvolvedores do *MySQL*, com a garantia de código aberto, consegue transformar dados em uma ampla gama de aplicativos, devido a sua robusteza, escalabilidade e rapidez (MariaDB, 2021).

O banco de dados MariaDB foi desenvolvido com um software de código aberto relacional, que fornece uma interface *SQL* para que os dados sejam acessados. Com suas versões atuais, as suas interfaces incluem também recursos de GIS (*Geographic Information System*) e JSON (*JavaScript Object Notation*).

Figura 13 – MariaDB Reserved Words (Keywords)

ACCESSIBLE	COLLATE	DEFAULT	FETCH	INFILE	LEFT
ADD	COLUMN	DELAYED	FLOAT	INNER	LIKE
ALL	CONDITION	DELETE	FLOAT4	INOUT	LIMIT
ALTER	CONSTRAINT	DELETE_DOMAIN_ID	FLOAT8	INSENSITIVE	LINEAR
ANALYZE	CONTINUE	DESC	FOR	INSERT	LINES
AND	CONVERT	DESCRIBE	FORCE	INT	LOAD
AS	CREATE	DETERMINISTIC	FOREIGN	INT1	LOCALTIME
ASC	CROSS	DISTINCT	FROM	INT2	LOCALTIMESTAMP
ASENSITIVE	CURRENT_DATE	DISTINCTROW	FULLTEXT	INT3	LOCK
BEFORE	CURRENT_ROLE	DIV	GENERAL	INT4	LONG
BETWEEN	CURRENT_TIME	DO_DOMAIN_IDS	GRANT	INT8	LONGBLOB
BIGINT	CURRENT_TIMESTAMP	DOUBLE	GROUP	INTEGER	LONGTEXT
BINARY	CURRENT_USER	DROP	HAVING	INTERSECT	LOOP
BLOB	CURSOR	DUAL	HIGH_PRIORITY	INTERVAL	LOW_PRIORITY
BOTH	DATABASE	EACH	HOUR_MICROSECOND	INTO	MASTER_HEARTBEAT_PERIOD
BY	DATABASES	ELSE	HOUR_MINUTE	IS	MASTER_SSL_VERIFY_SERVER_CERT
CALL	DAY_HOUR	ELSEIF	HOUR_SECOND	ITERATE	MATCH
CASCADE	DAY_MICROSECOND	ENCLOSED	IF	JOIN	MAXVALUE
CASE	DAY_MINUTE	ESCAPED	IGNORE	KEY	MEDIUMBLOB
CHANGE	DAY_SECOND	EXCEPTEXISTS	IGNORE_DOMAIN_IDS	KEYS	MEDIUMINT
CHAR	DEC	EXIT	IGNORE_SERVER_IDS	KILL	MEDIUMTEXT
CHARACTER	DECIMAL	EXPLAIN	IN	LEADING	MIDDLEINT
CHECK	DECLARE	FALSE	INDEX	LEAVE	+99 RESERVED WORDS...

Fonte: Adaptado de MariaDB (2021).

As *keywords* (palavras-chave) são excelentes formas de consulta que podem ser aplicadas em um banco de dados, métricas como políticas de retenção de dados são utilizadas para coleta, e bastante eficientes em casos de solução de problemas envolvendo um DBA (*Database Administrator*) verificando a aplicação, relatados na Figura 13.

4.3.2 Oracle Database

Os produtos Oracle proporcionam para os clientes desempenho elevado, optimização de custos e uma arquitetura extremamente completa para que seja um ambiente seguro para se trabalhar com aplicações (ORACLE, 2021).

A tarefa fundamental de um banco de dados é a segurança no momento de armazenar as informações. Segundo a Oracle, é possível descrever a estrutura de armazenamento em dois pontos:

a) Estrutura física;

b) Estrutura lógica.

A estrutura de armazenamento físico é apenas o arquivo que armazena os dados. Quando se é executado comandos como por exemplo para a criação de um novo banco de dados, os arquivos de maior importância a seguintes são criados:

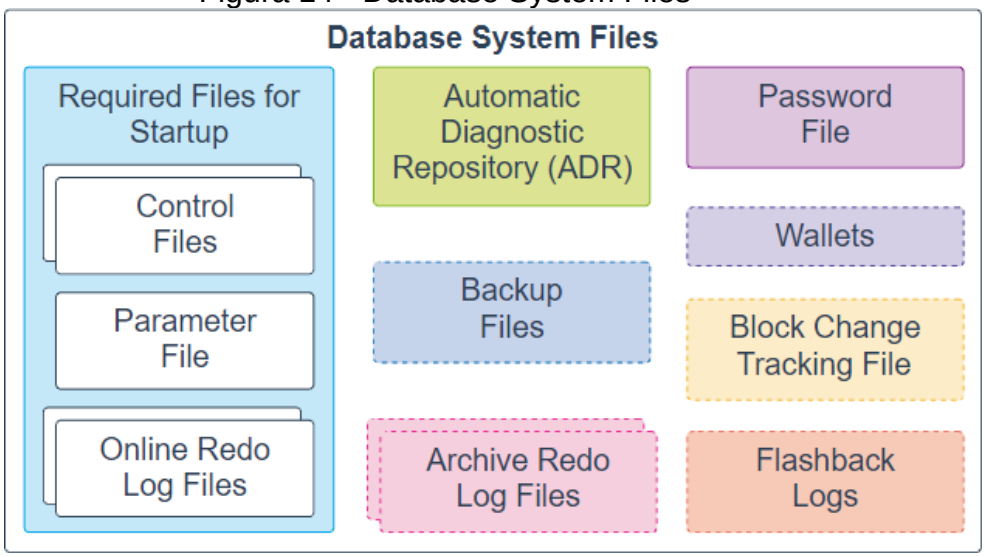
a) Data Files: Muito utilizado para índices e tabelas;

b) Control Files: Contém metadata e localização de arquivos;

c) Redo Log Files: Informações de logs e alterações.

Além dessa criação, arquivos como parâmetros de rede, senhas, backups, entre outros também são criados, sendo possível ver na Figura 14.

Figura 14 - Database System Files

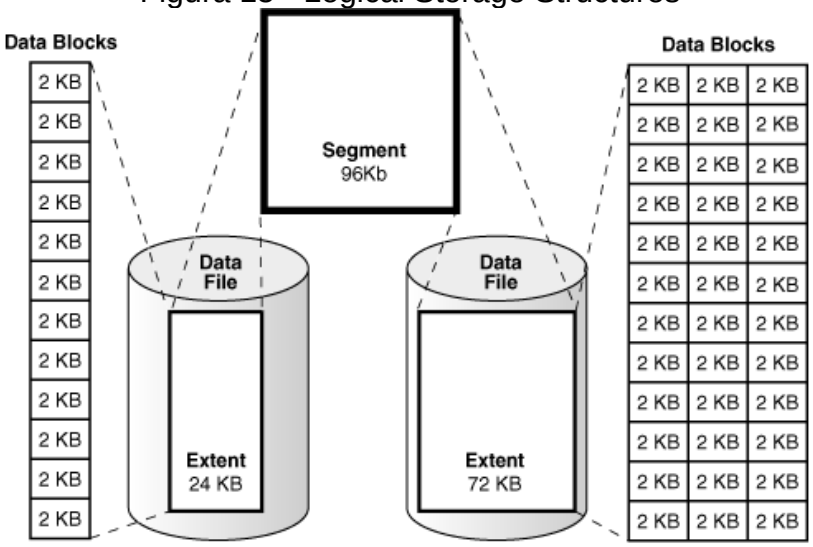


Fonte: https://www.oracletutorial.com/oracle-administration/oracle-database-architecture/ Acesso em: 28 maio 2021

A estrutura de armazenamento lógico está voltada para o controle do espaço em disco, sendo composta pelos seguintes elementos, juntamente com a Figura 15 para uma melhor compreensão.

a) Data blocks: Condiz com o número de bytes no disco;
b) Extents: Número específico para armazenar um tipo particular de informação;
c) Segments: É um conjunto de extents para armazenar objetos em uma tabela;
d) Tablespaces: É um contêiner que possui pelo menos um datafile;

Figura 15 - Logical Storage Structures



Fonte: https://www.oracletutorial.com/oracle-administration/oracle-database-architecture/ Acesso em: 28 maio 2021

4.4 ESCOLHA DA FERRAMENTA DE VISUALIZAÇÃO

Com as ferramentas de visualização é possível converter dados distintos em um formato coerente para que possam ser analisados e facilmente interpretados. Além de ser possível a customização de gráficos relacionados às métricas, é uma ótima forma de otimização de processos (Gomes, 2019).

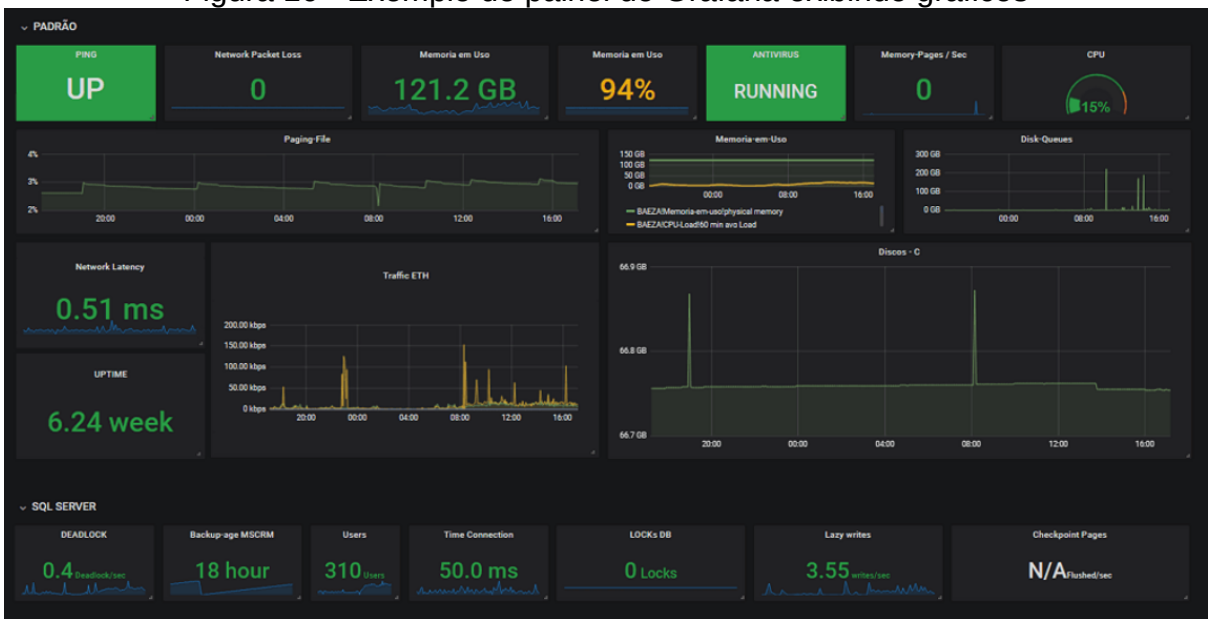
4.4.1 Grafana

O Grafana é uma ferramenta *open-source* que possui suporte para inúmeros bancos de dados, é altamente utilizado em setores empresariais devido a sua alternativa para visualização de alertas em tempo real. Com isso, é possível realizar a construção de painéis customizáveis que além de facilitarem a interpretação de alertas, colaboram com o conceito de alta disponibilidade da aplicação (Gomes, 2019).

Segundo Gomes (2019), além de ser extremamente útil no ambiente empresarial, o Grafana tem características muito positivas, dentre elas:

a) Estrutura amiga: Fácil instalação e muito eficiente;
b) Comunidade colaborativa: Sendo código aberto, possui grande número de otimizações.

Figura 16 - Exemplo de painel do Grafana exibindo gráficos



Fonte: https://www.opservices.com.br/grafana/ Acesso em: 28 maio 2021

Como pode ser visto na Figura 16, o Grafana é uma excelente ferramenta de visualização. Após a criação dos templates de alertas e coletas, a consequência é o auxílio na tomada de decisões ou para avaliar o status operacional do ambiente empresarial.

4.4.2 Telegram

O Telegram é uma ferramenta de mensagem instantânea baseada na tecnologia *cloud*. O aplicativo está disponível para *smartphones,* e é uma excelente ferramenta para ter integração com serviços de monitoramento, pois possui diversas funcionalidades de API que visam a alta disponibilidade.

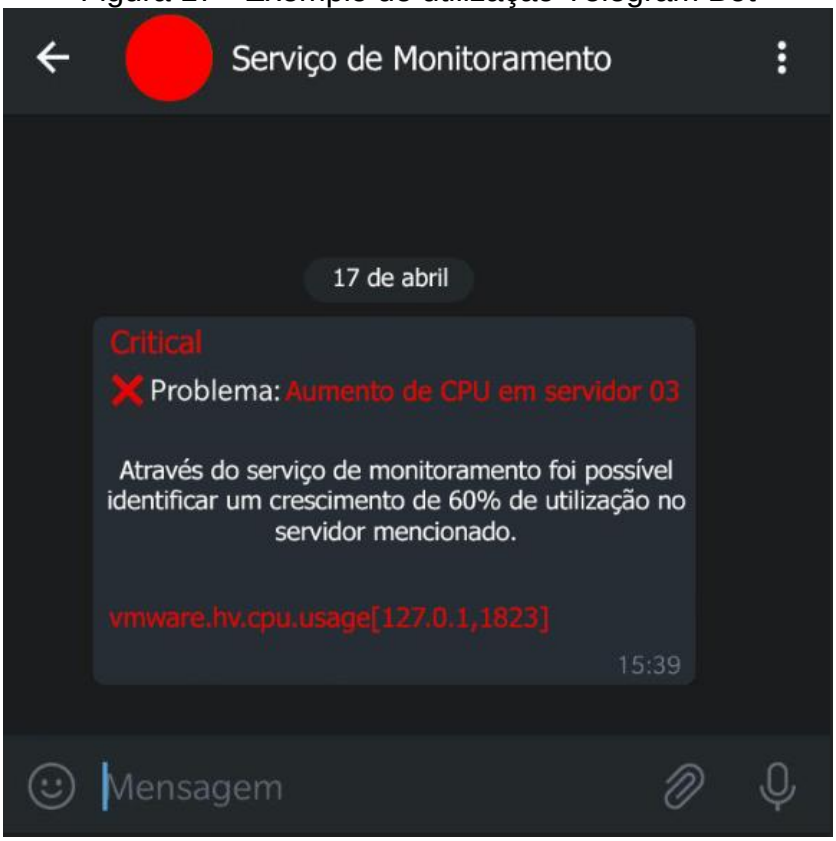
O crescimento do Telegram é notável, superando o número de *downloads* do aplicativo *WhatsApp* no mês de janeiro de 2021 (CNN, 2021). Por conta desse crescimento, notou-se boas alternativas para utilização em diferentes plataformas e situações através do *Telegram Bot*.

O *Telegram Bot* é uma conta especial que pode ser configurada em diferentes tipos de plataformas, nenhum outro número de telefone precisa ser definido. Essas contas são usadas como uma interface para o código em execução em algum lugar do servidor (Telegram, 2021). Com a configuração certa referente à aplicação, é

possível utilizar esse tipo de tecnologia para facilitar a administração de serviços de monitoramento, tanto quanto fazer o envio de alertas em tempo real para que as correções (caso necessário) tenham um tempo de resolução rápido.

A Figura 17 é um exemplo de monitoramento em tempo real utilizando a inteligência do *Telegram Bot.*

Figura 17 - Exemplo de utilização Telegram Bot



Fonte: Autoria própria

Embora o Telegram seja um recurso opcional, é de grande importância para a realização deste trabalho. Através dele, foi possível configurar um ambiente de infraestrutura capaz de proporcionar alertas em tempo real e configurar, mediante a aplicação, formas de resolver as adversidades sem a necessidade de intervenção humana.

4.5 VALIDAÇÕES

Com base nas ferramentas estudadas no capítulo 4, os *softwares* foram analisados e comparados para que pudessem ser definidos antecipadamente neste

trabalho. Com isso, os parâmetros para a escolha das ferramentas foram preservados e a escolha foi definida através das comparações feitas nos Quadros 1, 2 e 3.

Quadro 1 - Parâmetros para escolha da ferramenta coletora

Parâmetros	**Nagios**	**Zabbix Proxy**
Compatibilidade com distribuições Linux	Validada	Validada
Gratuidade	Validada	Validada
Licença *open-source*	Validada	Validada
Estrutura de métricas	Validada	Validada
Intervalo de tempo customizável	Validada	Validada
Continuidade com versões atualizadas	Validada	Validada

Fonte: Autoria própria.

Embora todos os parâmetros estejam validados em ambas ferramentas, cabe-se diferenciar os procedimentos de instalação que estão nos documentos de cada um. Apesar de o *software* Nagios ter uma instalação mais fácil, percebeu-se uma carência de informações, pois caso ocorram erros durante a instalação, seria interessante um auxílio nesse sentido.

A ferramenta do Zabbix foi considerada satisfatória, embora mais complexa para o entendimento. O manual de instalação e suporte para possíveis erros estão mais adequados, consequentemente deixando o usuário mais confortável em momentos extremos. Por conta desses detalhes, a ferramenta escolhida foi o Zabbix.

A respeito da escolha da ferramenta de armazenamento, todas as características dos *softwares* analisados atenderam os critérios de avaliação. Contudo, é válido ressaltar que a usabilidade do banco de dados MariaDB se destacou devido a sua grande utilidade no mercado, além de ter melhores recursos além de ter instalações simples e bem estruturadas, que é um ponto extremamente importante e crítico para empresas com aplicativos mais sensíveis, grandes quantidades de dados e números relevantes de usuários simultâneos.

Quadro 2 - Parâmetros para escolha da ferramenta de armazenamento

Parâmetros	**MariaDB**	**Oracle Database**
Compatibilidade com distribuições Linux	Validada	Validada
Gratuidade	Validada	Validada
Licença *open-source*	Validada	Validada
Retenção de dados customizável	Validada	Validada

Fonte: Autoria própria.

Com relação às ferramentas de visualização, tanto o Telegram (notificação através do *smartphone*) quanto o Grafana (visualização através do navegador) são elementos primordiais para a realização deste trabalho. Além de cumprirem todos os parâmetros propostos no capítulo, suas configurações são totalmente adaptáveis ao tema proposto.

Quadro 3 - Parâmetros para escolha da ferramenta de visualização

Parâmetros	**Grafana**	**Telegram**
Compatibilidade com distribuições Linux	Validada	Validada
Gratuidade	Validada	Validada
Licença open-source	Validada	Validada
Compatibilidade Interface Web	Validada	Validada
Envio de alertas (Apto a comunicar)	Validada	Validada
Compatibilidade com coleta de dados	Validada	Validada

Fonte: Autoria própria.

4.5.1 Monitoramento funcional

Segundo Gaidargi (2019), um monitoramento funcional é responsável por avaliar o desempenho e a disponibilidade de um ambiente monitorado. Através do Quadro 4, é possível ver as exigências necessárias para o êxito do funcionamento, tanto quanto as características fundamentais para que o mesmo ocorresse.

Quadro 4 - Exigências e características do monitoramento funcional

Exigências	**Características**
Visualização da saúde dos ambientes	O usuário deve ter acesso a ferramenta através do browser de internet, e poderá desfrutar das utilidades.
Entendimento claro e compreensível dos alertas	O ambiente de monitoramento deve fornecer dados capazes do usuário entender e atuar, caso necessário.
Alta disponibilidade da aplicação e do monitoramento	O ambiente de monitoramento deve coletar dados, mantendo a sua função de controle de métricas essenciais para um ambiente de infraestrutura.

Fonte: O Autor.

4.5.2 Métricas e alertas

O Quadro 5 é responsável por expor os detalhes sobre o que diz respeito a métricas e alertas. Através da configuração do ambiente, e das diferentes criações de painéis via Grafana, pôde ser viável o acompanhamento de alertas e também do histórico de incidentes.

Quadro 5 - Exigências e características de métricas e alertas

Exigências	**Características**
Envio de alertas via Telegram	O ambiente deve estar configurado para que seja possível fazer o envio de alertas através da plataforma Telegram
Categorização de alertas via severidade	A plataforma deve estar adequada a personalização de severidades de alertas conforme importância da arquitetura
Recurso opcional para correção de alertas.	A plataforma deve proporcionar o recurso de resolução automática de alertas, para que seja possível manter a alta disponibilidade da aplicação sem intervenção humana.

Fonte: O Autor.

5 DESENVOLVIMENTO

A inclusão das ferramentas de código aberto Zabbix, MariaDB e Grafana, escolhidas anteriormente através de critérios selecionados foi feita com o propósito de disponibilizar um modelo de monitoramento de servidores Linux com o uso de ferramentas gratuitas.

Para que fosse possível a comunicação entre as ferramentas, a integração foi estruturada através do transporte de dados entre as mesmas. Segundo Vanover (2014), o recurso *Internal Virtual Switch no Hyper-V* permite que a comunicação interna entre as máquinas virtuais possa ser estabelecida somente se estiverem relacionadas com o mesmo *host*, pois outras formas de comunicação, tais como a comunicação privada, não estão aptas a se comunicarem com outras máquinas que estejam fora do *host* mencionado.

5.1 ELABORAÇÃO DO AMBIENTE

Abaixo, é possível visualizar os *softwares* que foram utilizados na máquina virtual hospedeira, juntamente com as suas versões:

- Oracle Linux versão 8.4;
- MariaDB versão 10.6.4;
- Zabbix versão 5.4.6;
- Grafana versão 8.2.2.

Na máquina virtual coletável, os recursos que foram utilizados são:

- Oracle Linux versão 8.4;
- MariaDB versão 10.6.4;
- Zabbix versão 5.4.6 (*agent*);

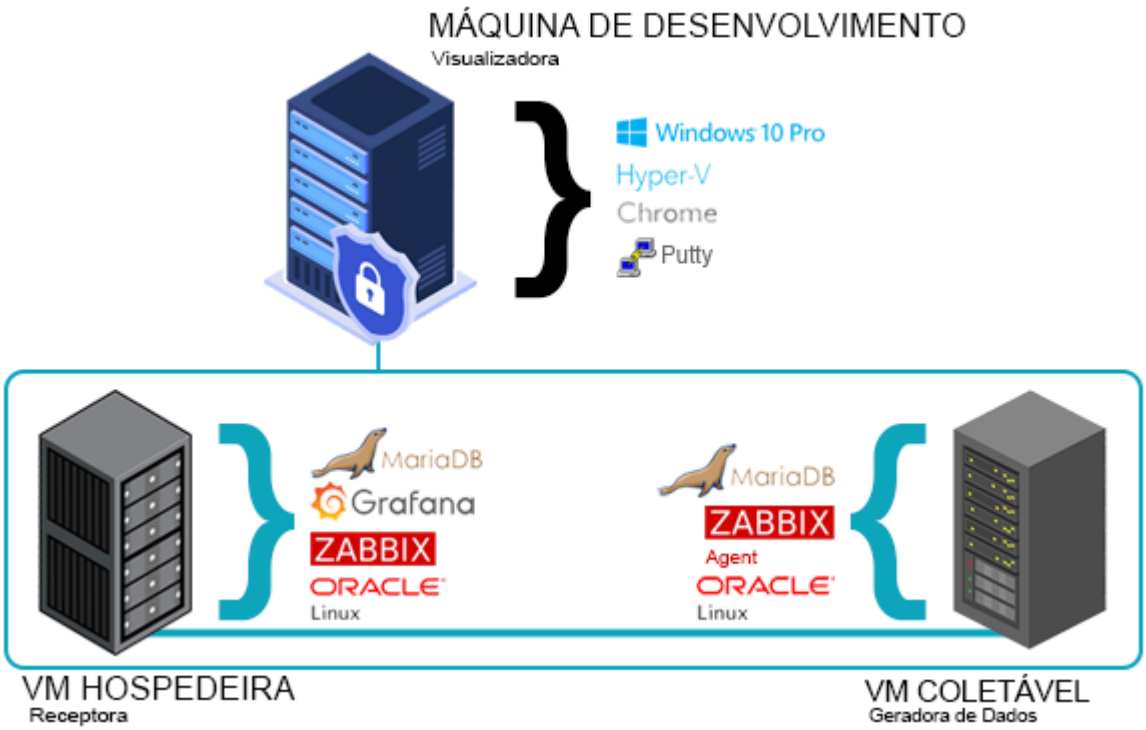
Na máquina do desenvolvimento, os recursos que foram utilizados são:

- Windows 10 Professional;
- Hyper-V Versão: 10.0.19041.1;
- Navegador Google Chrome;
- Putty Versão 0.76.

Conforme a Figura 18, existem duas máquinas virtuais: uma responsável pela geração de alertas (VM Coletável), e uma responsável pelo recebimento de dados (VM Hospedeira). Toda a arquitetura será executada através do servidor de desenvolvimento, na qual é responsável pelo provimento do ambiente e de componentes essenciais, tais como: memória e armazenamento para que seja possível ter a criação do ambiente.

No gráfico da Figura 18, é possível ver todos os *softwares* escolhidos para que fosse possível a criação do ambiente proposto.

Figura 18 - Comunicação das ferramentas



Fonte: Autoria própria.

.

5.2 INSTALAÇÃO DAS FERRAMENTAS

Para que a criação pudesse ser realizada, as instalações foram feitas conforme os parâmetros definidos como padrão. Da mesma forma, ajustes para comunicação entre elementos da rede, visando critérios de segurança e performance na hora da coleta de dados.

5.2.1 Instalação e parametrização do MariaDB

Primeiramente foi feita a instalação do banco de dados MariaDB. Nesse momento, o repositório da aplicação foi adicionado (*mariadb.repo*), seguindo com os comandos de instalação oriundos da distribuição Oracle Linux. A instalação teve continuidade com base no procedimento documentado e homologado da empresa desenvolvedora (através do comando *YUM*), criação de *tablespaces*, privilégios de administrador (*root*) para compatibilidade, escalabilidade e liberação de portas (também com foco em *http* para comunicação web). Na Figura 19 é possível ver as estatísticas da rede através do comando *netstat,* mostrando as conexões *TCP* e conexões por protocolo de acordo com o nome de cada programa e serviço.

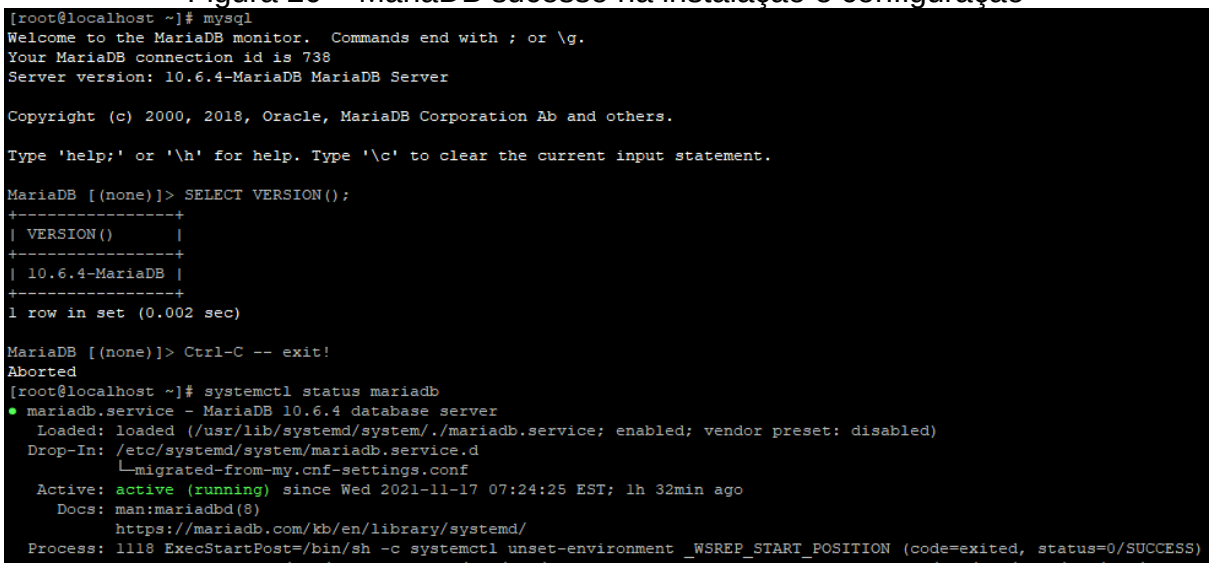
Figura 19 – Estatísticas da rede netstat -tlnp

```
[root@localhost ~]# netstat -tlnp
Conexões Internet Ativas (sem os servidores)
Proto Recv-Q Send-Q Endereço Local           Endereço Remoto         Estado      PID/Program name
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               OUÇA        824/mariadbd
tcp        0      0 0.0.0.0:22              0.0.0.0:*               OUCA        792/sshd
tcp6       0      0 :::3306                 :::*                    OUÇA        824/mariadbd
tcp6       0      0 :::80                   :::*                    OUÇA        796/httpd
tcp6       0      0 :::22                   :::*                    OUÇA        792/sshd
```

Fonte: Autoria própria.

Por fim, conforme a Figura 20, é possível ver que o serviço do banco de dados foi habilitado e iniciado na VM hospedeira (*active running*), tanto quanto na VM coletável, através do comando “*systemctl enable --now mariadb*”. Dessa forma, foi possível continuar o desenvolvimento do trabalho, uma vez que o banco de dados foi corretamente instalado e executado.

Figura 20 – MariaDB sucesso na instalação e configuração



Fonte: Autoria própria.

5.2.2 Instalação e parametrização do Zabbix

A instalação e parametrização do Zabbix nas máquinas virtuais sucedeu através de procedimentos de instalação por meio de gerenciadores de pacotes (tais como *YUM*, *APT*, *DNF*, conforme distribuição), sendo instalado o Zabbix *server*, Zabbix *frontend,* Zabbix *apache* e o Zabbix *agent*.

Considerando a instalação dos pacotes finalizados, foi criado um novo banco de dados *MySQL*, com suporte a caracteres *UTF-8* (codificação de caracteres mais comum da *World Wide Web*), em seguida, houve a criação de um usuário (com privilégios de administrador) ao servidor Zabbix, garantindo todos os privilégios ao banco de dados instalado, conforme a Figura 21.

Figura 21 – Criação e configuração do banco de dados

```
mysql> create database zabbix character set utf8 collate utf8_bin;
mysql> create user zabbix@localhost identified by 'admin123';
mysql> grant all privileges on zabbix.* to zabbix@localhost;
mysql> quit;
```

Fonte: Autoria própria.

Logo após, foi necessário importar o esquema inicial e os dados, na qual foi gerado através da instalação do Zabbix um arquivo configurável, usando o comando *zcat* (para a descompactação), já que os dados estão comprimidos. Para que fosse possível o servidor Zabbix usar o banco de dados instalado, foi preciso definir a senha

do banco de dados no arquivo de configuração do servidor Zabbix chamado de *zabbix_server.conf*, conforme demonstrado na Figura 22.

Figura 22 – Parametrizações no arquivo *zabbix_server.conf*.

```
DBUser=zabbix

### Option: DBPassword
#       Database password.
#       Comment this line if no password is used.
#
# Mandatory: no
# Default:
DBPassword=admin123
```

Fonte: Autoria própria.

Por fim, ao final da parametrização, foi possível a utilização do *frontend* do Zabbix através do navegador. Para que fosse possível a coleta de dados dos servidores, foi realizada a configuração dos *hosts* das máquinas virtuais. Foi definido um intervalo de 10 segundos entre as coletas. Dessa forma, foi possível garantir a visualização gráfica, uma vez que os dados são mais detalhados, sendo coletados pelo *agent* do Zabbix.

Além do mais, em cada *host* foi vinculado a *template Linux by Zabbix Agent*, visando os principais *plugins* referentes a infraestrutura do sistema operacional. Dentre eles, "*Linux CPU by Zabbix Agent"* para coleta de métricas de processamento, "*Linux memory by Zabbix Agent"* para métricas com relação a memória *RAM,* "*uptime"* para métricas de tempo em relação ao último *boot*, e "*Linux network by Zabbix Agent"* para métricas de rede, conforme é possível ver na Figura 23.

Figura 23 – Configuração dos *hosts* e *templates*

Fonte: Autoria própria.

5.2.3 Instalação e parametrização do Grafana

Para que fosse possível fazer a instalação e parametrização da ferramenta de visualização Grafana, também foram seguidos os manuais e documentos de homologação disponibilizados. O primeiro passo foi instalar o repositório do Grafana (*grafana.repo*) na máquina virtual hospedeira, fazendo a parametrização e configuração do arquivo conforme a Figura 24.

Figura 24 – Parametrização Grafana

```
[grafana]
name=grafana
baseurl=https://packages.grafana.com/oss/rpm
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://packages.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

Fonte: Autoria própria.

A instalação e parametrização do Grafana também ocorreu através de procedimentos habituais de instalação por meio de gerenciadores de pacotes (*YUM*), habilitando o serviço da aplicação e também liberando a porta de rede 3000 (exibido na Figura 25), para que fosse possível dar continuidade a configuração através da capacidade da interface gráfica.

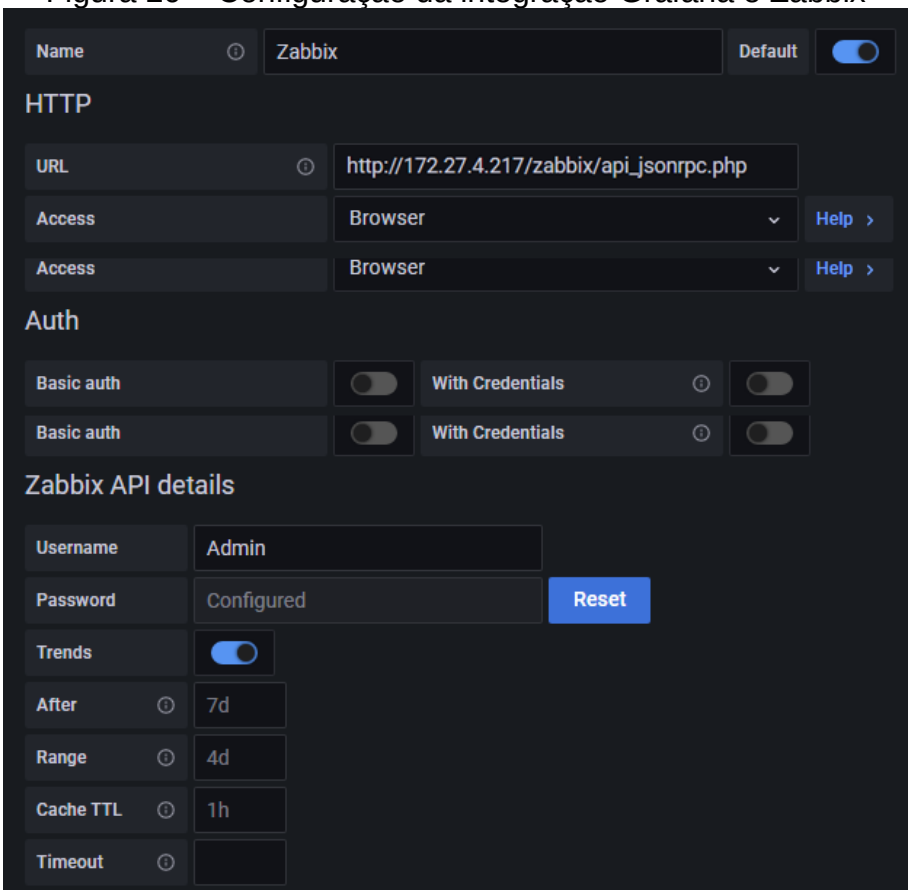
Figura 25 – Liberação *firewall* porta 3000

```
[root@localhost ~]# firewall-cmd --zone=public --add-port=3000/tcp --permanent
success
```

Fonte: Autoria própria.

Na sequência, foi preciso fazer a integração entre o Grafana e o Zabbix através da interface gráfica. Por meio da parametrização de usuários, senhas, portas e IP foi possível ter a compatibilidade entre as plataformas envolvidas, conforme pode ser visto na tela de *data source* representado na Figura 26.

Figura 26 – Configuração da integração Grafana e Zabbix



Fonte: Autoria própria.

Por fim, a instalação e a parametrização do Grafana foram feitas com sucesso, permitindo a compatibilidade entre ferramentas e a disponibilidade de criação de gráficos em tempo real, referentes aos alertas que o *Zabbix Agent* possa coletar.

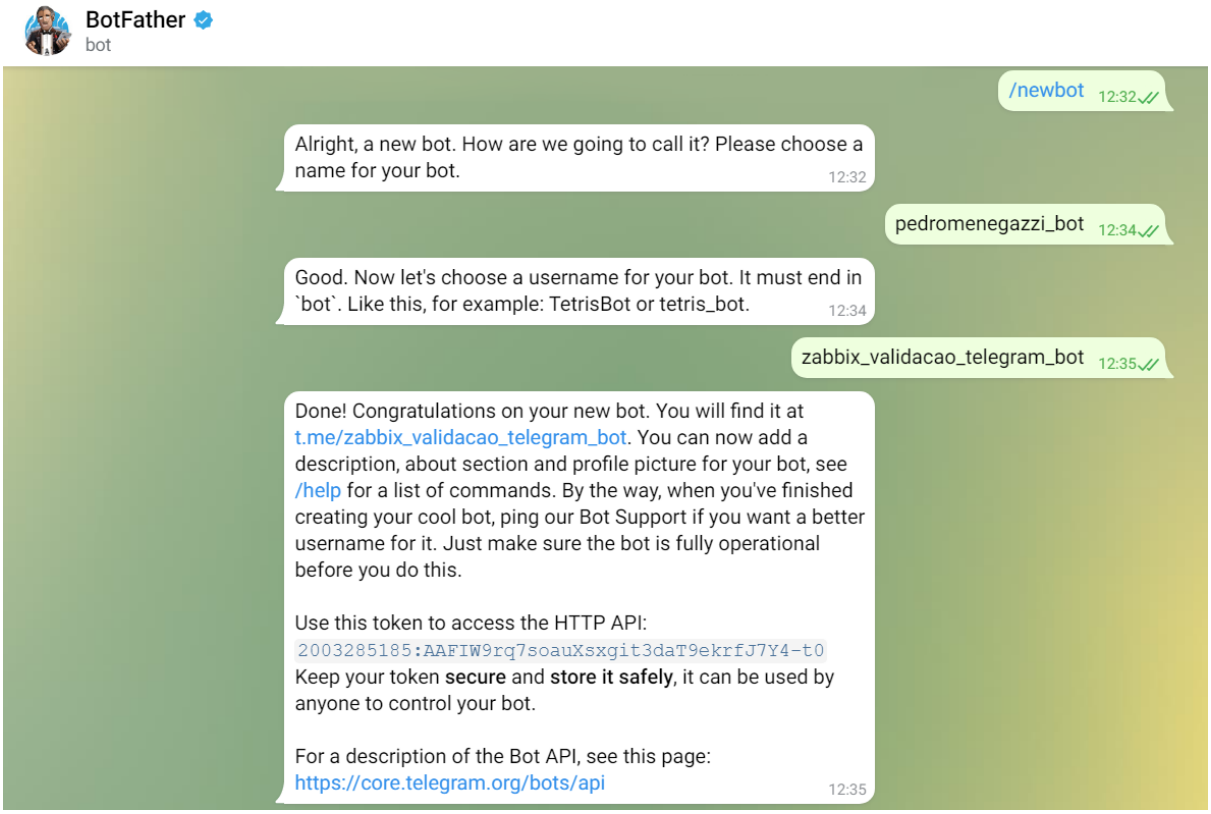
5.2.4 Instalação e parametrização do Telegram

Para que pudesse ser parametrizado o recurso de recebimento de alertas via smartphone, foi necessário a configuração do Telegram. Na primeira etapa, foi feito o *download* e criação da conta através do *smartphone*, utilizando o número pessoal e a validação necessária (através do próprio *software*) para confirmar a legitimidade da conta e informações pessoais.

Logo após, foram seguidos os manuais e documentos de homologação disponibilizados. O primeiro passo referente a configuração do Telegram foi registrar um novo Telegram Bot através do “@BotFather”, que se caracteriza por ser um *bot* que permita ao usuário a criação e gerenciamento dentro da própria plataforma do

Telegram. Nessa etapa, foi feita a criação de um novo *bot* com o comando */newbot*. Na sequência, houve também a definição do nome (*pedromenegazzi_bot*) e usuário (*zabbix_validacao_telegram_bot*) do *bot* criado, como é possível ver na Figura 27.

Figura 27 – Configuração e parametrização do *@BotFather*

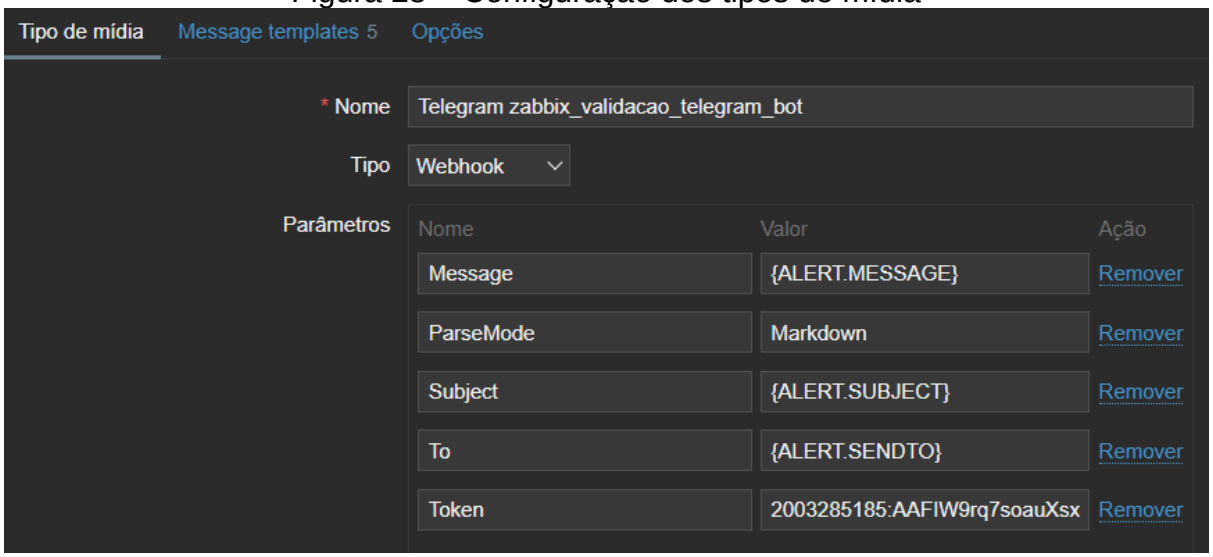


Fonte: Autoria própria.

Após a criação do *bot,* foi disponibilizado um *token,* cuja principal função é de controle administrador do *bot* criado. A homologação do Telegram prioriza a entrega de mensagens de forma nominal, ou seja, para cada usuário específico, foi necessário consultar o *ID* da conta criada. Para obter essa informação utilizou-se o "*IDBot*", na qual revelou o *ID* característico da conta.

Com os *bots* configurados e parametrizados no Telegram, foi necessário fazer a configuração dos tipos de mídias dentro da interface gráfica da plataforma Zabbix. As informações definidas foram o nome da mídia, os parâmetros de coleta de mensagem {*alert.message*}, título {*alert.subject*}, destino {*alert.sendto*} e o *token* fornecido pelo *bot* criado no telegrama para controle e administração, como é possível analisar na Figura 28.

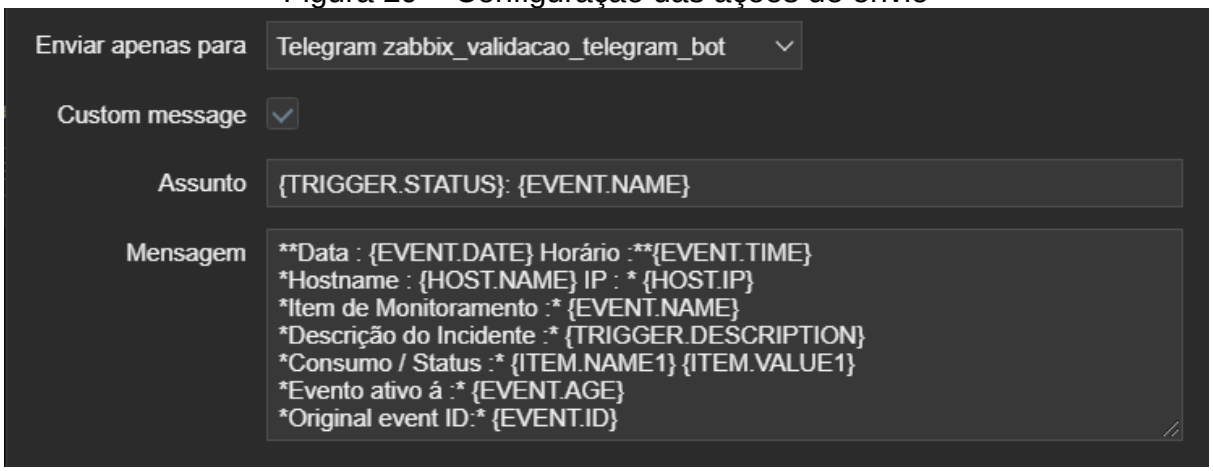
Figura 28 – Configuração dos tipos de mídia



Fonte: Autoria própria.

No que se refere ao envio de notificações e configurações de alertas, foi preciso criar uma ação específica dentro da interface gráfica do Zabbix, para que no momento que acionada, envie uma mensagem instantaneamente para o *bot* criado anteriormente. A configuração foi criada com o objetivo de que todo alerta coletado com severidade maior ou igual a "*Desastre*" disparasse um comando com as variáveis obtidas através do *Zabbix Agent* sobre data do evento, horário, *hostname, IP*, item de monitoramento, descrição do incidente, consumo/status, evento do ativo e o seu *ID,* conforme é possível ver na Figura 29.

Figura 29 – Configuração das ações de envio



Fonte: Autoria própria.

Por fim, foi necessário fazer o vínculo da conta de administrador do Zabbix com as credenciais e privilégios do *bot* no Telegram. Após essa configuração, tornou-se possível o recebimento instantâneo de alertas através das plataformas.

6 VALIDAÇÃO E RESULTADOS

Logo após o desenvolvimento do ambiente de monitoramento e de suas ferramentas de visualização, foram realizados testes com base no capítulo 4.5, referente as validações da solução. O planejamento dos processos foi definido visando os seguintes objetivos:

- ➢ *Visualização da saúde dos ambientes;*
- ➢ *Entendimento claro e compreensível dos dados;*
- ➢ *Alta disponibilidade da aplicação e do monitoramento;*
- ➢ *Envio de alertas via Telegram;*
- ➢ *Categorização de alertas via severidade;*
- ➢ *Recurso opcional para correção de alertas.*

6.1 CASOS DE TESTES

Nas subseções a seguir, serão apresentados seis casos de testes: Visualização da saúde dos ambientes, entendimento claro e compreensível dos dados, alta disponibilidade da aplicação e do monitoramento, envio de alertas via Telegram, categorização de alertas via severidade e recurso opcional para correção de alertas.

6.1.1 Caso de teste “Visualização da saúde dos ambientes”

O primeiro teste, que está sendo representado no Quadro 6, foi “Visualização da saúde dos ambientes”.

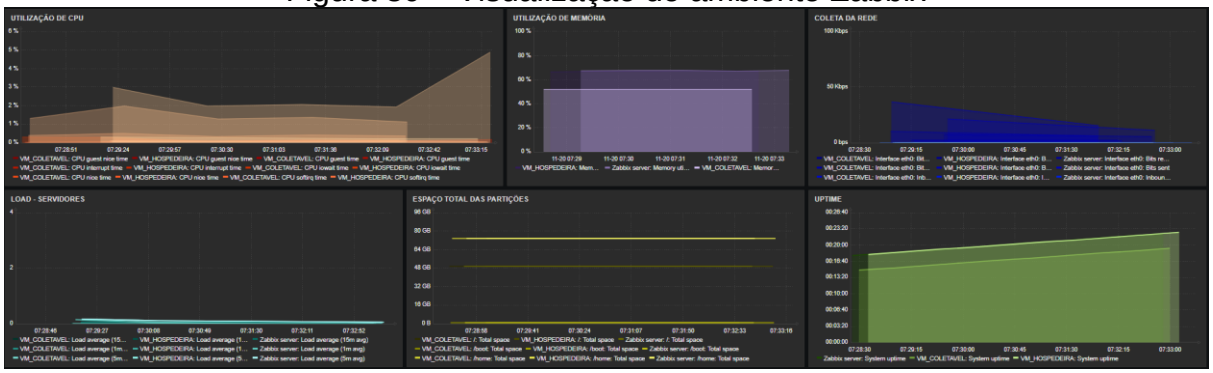
Quadro 6 - Visualização da saúde dos ambientes

ID	NR -1
Nome	Visualização da saúde dos ambientes
Ator(es)	Usuário administrador
Pré-Condições	O usuário deve ter acesso ao Grafana e ao Zabbix.
Procedimentos	I. O usuário acessa o *browser*; II. O usuário seleciona os endereços das plataformas; III. O usuário preenche os dados de autenticação; IV. O usuário acessa o setor de Monitoramento no Zabbix; V. O usuário acessa o setor de *Dashboard* no Zabbix; VI. O usuário acessa o setor de Pesquisa no Grafana; VII. O usuário acessa o setor de *Dashboard* no Grafana
Pós-Condições	O usuário tem acesso à saúde dos ambientes.

Fonte: O Autor.

A visualização da saúde do ambiente do Zabbix foi feita utilizando o Google Chrome, conforme Figura 30. Na figura é possível ver que os gráficos exibidos revelam a saúde do ambiente monitorado, tanto quando métricas importantes referentes as *dashboards* customizadas.

Figura 30 – Visualização do ambiente Zabbix



Fonte: Autoria própria.

Da mesma forma, é possível ver, conforme a Figura 31, a visualização da saúde do ambiente do Grafana. Os gráficos exibidos revelam a saúde do monitoramento das máquinas virtuais e também a customização de painéis conforme necessidade do usuário administrador.

Figura 31 – Visualização do ambiente Grafana

Fonte: Autoria própria.

6.1.2 Caso de teste “Entendimento claro e compreensível dos alertas”

O segundo teste, que pode ser observado no Quadro 7, foi “Entendimento claro e compreensível dos alertas”.

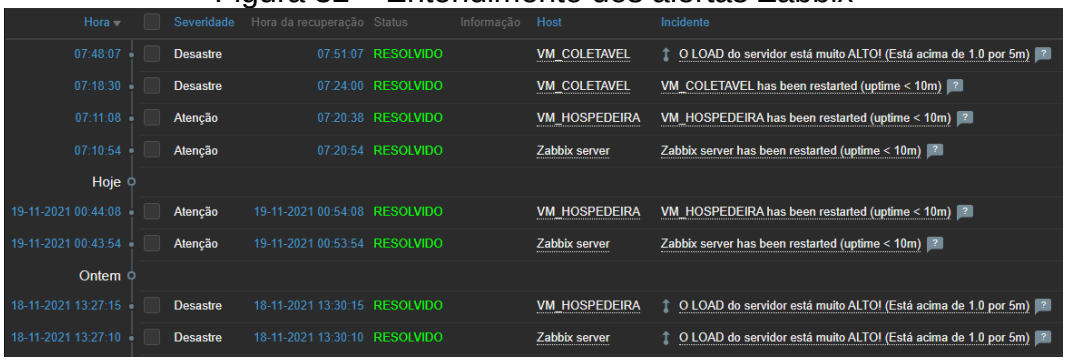
Quadro 7 - Entendimento claro e compreensível dos alertas

ID	NR - 2
Nome	Entendimento claro e compreensível dos alertas
Ator(es)	Usuário administrador
Pré-Condições	O usuário deve ter entendimento dos alertas coletados via Zabbix
Procedimentos	I. O usuário abre o Zabbix via *browser*; II. O usuário seleciona o setor “Incidentes”; III. O usuário filtra os incidentes através da aba “Histórico”.
Pós-Condições	O usuário tem o entendimento dos alertas coletados em tempo real, informação de status, severidade, incidente e duração.

Fonte: O Autor.

Os procedimentos foram feitos utilizando o Google Chrome, resultado em um acesso livre de erros. Conforme a visualização, na Figura 32, é possível ver alertas coletados através do ambiente Zabbix os últimos dois dias A figura mostra o entendimento da coleta com informações valiosas referentes a tempo, status, severidade e duração do alerta individual.

Figura 32 – Entendimento dos alertas Zabbix



Hora		Severidade	Hora da recuperação	Status	Informação	Host	Incidente
07:48:07		Desastre	07:51:07	RESOLVIDO		VM_COLETAVEL	O LOAD do servidor está muito ALTO! (Está acima de 1.0 por 5m)
07:18:30		Desastre	07:24:00	RESOLVIDO		VM_COLETAVEL	VM_COLETAVEL has been restarted (uptime < 10m)
07:11:08		Atenção	07:20:38	RESOLVIDO		VM_HOSPEDEIRA	VM_HOSPEDEIRA has been restarted (uptime < 10m)
07:10:54		Atenção	07:20:54	RESOLVIDO		Zabbix server	Zabbix server has been restarted (uptime < 10m)
Hoje							
19-11-2021 00:44:08		Atenção	19-11-2021 00:54:08	RESOLVIDO		VM_HOSPEDEIRA	VM_HOSPEDEIRA has been restarted (uptime < 10m)
19-11-2021 00:43:54		Atenção	19-11-2021 00:53:54	RESOLVIDO		Zabbix server	Zabbix server has been restarted (uptime < 10m)
Ontem							
18-11-2021 13:27:15		Desastre	18-11-2021 13:30:15	RESOLVIDO		VM_HOSPEDEIRA	O LOAD do servidor está muito ALTO! (Está acima de 1.0 por 5m)
18-11-2021 13:27:10		Desastre	18-11-2021 13:30:10	RESOLVIDO		Zabbix server	O LOAD do servidor está muito ALTO! (Está acima de 1.0 por 5m)

Fonte: Autoria própria.

6.1.3 Caso de teste “Alta disponibilidade da aplicação e do monitoramento”

O terceiro teste, que pode ser observado no Quadro 8, foi “Alta disponibilidade da aplicação e do monitoramento”.

Quadro 8 - Alta disponibilidade da aplicação e do monitoramento

ID	NR - 3
Nome	Alta disponibilidade da aplicação e do monitoramento
Ator(es)	Usuário administrador
Pré-Condições	O usuário deve ter alta disponibilidade da aplicação e do monitoramento
Procedimentos	I. O usuário acessa o Grafana via *browser*; II. O usuário seleciona o setor de *Dashboards*; III. O usuário analisa informações do gráfico de *uptime*.
Pós-Condições	O usuário tem um ambiente de alta disponibilidade.

Fonte: O Autor.

Os procedimentos foram executados durante um intervalo de cinco horas ininterruptas, conforme é possível ver na Figura 33. Após o tempo estabelecido, o ambiente de monitoramento se manteve com funcionamento exemplar e não apresentou quedas e interrupções. É importante ressaltar que o tempo determinado foi um intervalo ajustado para análise de desempenho das plataformas.

Figura 33 – Disponibilidade do monitoramento

Fonte: Autoria própria.

6.1.4 Caso de teste “Envio de alertas via Telegram”

O quarto teste, que pode ser observado no Quadro 9, foi “Envio de alertas via Telegram”.

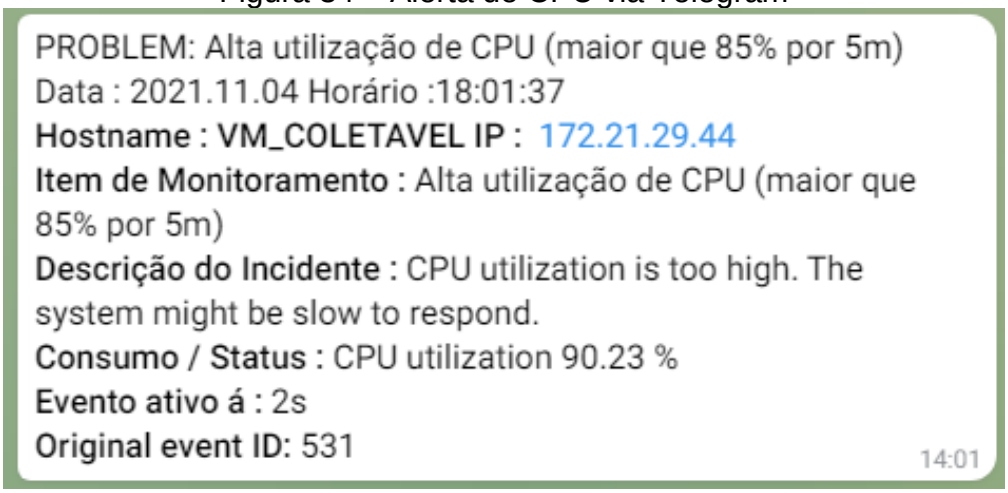
Quadro 9 - Envio de alertas via Telegram

ID	NR - 4
Nome	Envio de alertas via Telegram
Ator(es)	Usuário administrador
Pré-Condições	O usuário deve receber alertas em tempo real via Telegram
Procedimentos	I. O usuário recebe uma notificação via *smartphone*; II. O usuário acessa o grupo *bot* criado; III. O usuário tem o detalhamento do alerta recebido.
Pós-Condições	O usuário consegue ter o recurso de visualização de alertas em tempo real via *smartphone*.

Fonte: O Autor.

Os procedimentos tiveram a sua execução com alertas gerados através de testes de estresse (*stress level*), na qual foi possível gerar alertas referentes ao sistema operacional das máquinas virtuais, conforme Figura 34. A partir do momento em que os testes de estresse começaram a serem feitos, as altas métricas coletadas de *CPU*, memória *RAM* e pacotes de rede, o grupo de mensagens do Telegram registrou o recebimento de alertas com as descrições referentes ao Capítulo 5.2.4.

Figura 34 – Alerta de CPU via Telegram



Fonte: Autoria própria.

6.1.5 Caso de teste “Categorização de alertas via severidade”

O quinto teste, que pode ser observado no Quadro 10, foi “Categorização de alertas via severidade”.

Quadro 10 - Categorização de alertas via severidade

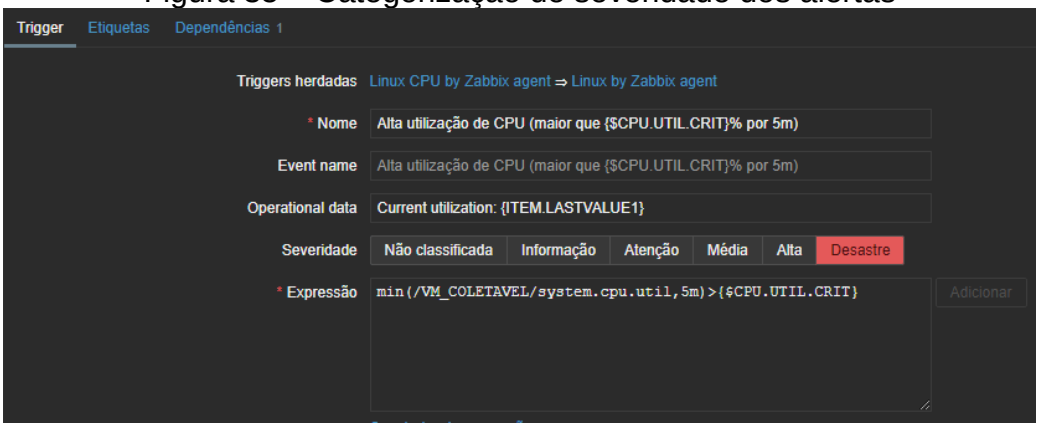
ID	NR - 5
Nome	Categorização de alertas via severidade
Ator(es)	Usuário administrador
Pré-Condições	O usuário deve ser capaz de categorizar a severidade do alerta desejado.
Procedimentos	I. O usuário acessa a interface do Zabbix via *browser*; II. O usuário acessa o setor de Hosts; III. O usuário seleciona o Host desejado; IV. O usuário acessa o setor de Triggers; V. O usuário seleciona o alerta desejado; VI. O usuário tem a opção de customizar conforme severidade do alerta.
Pós-Condições	O usuário tem a opção de customização do alerta conforme a necessidade do usuário administrador.

Fonte: O Autor.

Conforme o exemplo da Figura 35, foi possível fazer a customização de alertas via interface gráfica do Zabbix. No exemplo citado, houve a alteração de severidade

feita pelo usuário administrador referente a um alerta de alta utilização de CPU, definindo-o como “Desastre”. É importante ressaltar que no ambiente de monitoramento configurado, a severidade está associada diretamente com a parametrização do Telegram, cuja configuração foi definida para fazer a coleta de todos os alertas com severidade “Desastre”, impactando de modo direto, no que o usuário administrador opta pelo recebimento de mensagens em tempo real via *smartphone*.

Figura 35 – Categorização de severidade dos alertas



Fonte: Autoria própria.

6.1.6 Caso de teste “Recurso opcional para correção de alertas”

O sexto teste, conforme pode ser visto no Quadro 11, foi “Recurso opcional para correção de alertas”.

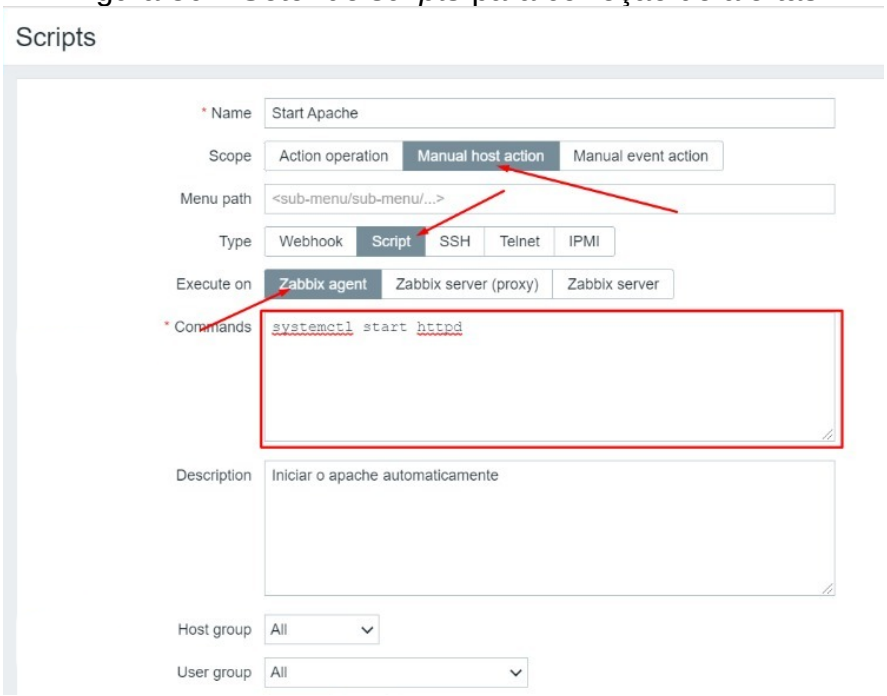
Quadro 11 – Recurso opcional para correção de alertas

ID	NR - 6
Nome	Recurso opcional para correção de alertas
Ator(es)	Usuário administrador
Pré-Condições	O monitoramento deve apresentar o recurso opcional de correção de alertas para o usuário administrador.
Procedimentos	I. O usuário acessa a interface do Zabbix via *browser*; II. O usuário acessa o setor de Administração; III. O usuário acessa o subsetor de Scripts; IV. O usuário faz a criação de um *script* desejado; V. O usuário acessa o setor de Configurações; VI. O usuário acessa o subsetor de Ações; VII. O usuário cria uma ação para disparo automático de alertas.
Pós-Condições	O usuário tem o recurso de correção automática de alertas via Zabbix.

Fonte: O Autor.

Conforme é possível ver na Figura 36, o usuário administrador tem a opção de criar *scripts* para disparo de comandos, conforme a necessidade e o objetivo do ambiente de monitoramento. No exemplo feito, foi criado um *script* de disparo sempre que o alerta de serviço do Apache não estiver disponível na máquina virtual coletável.

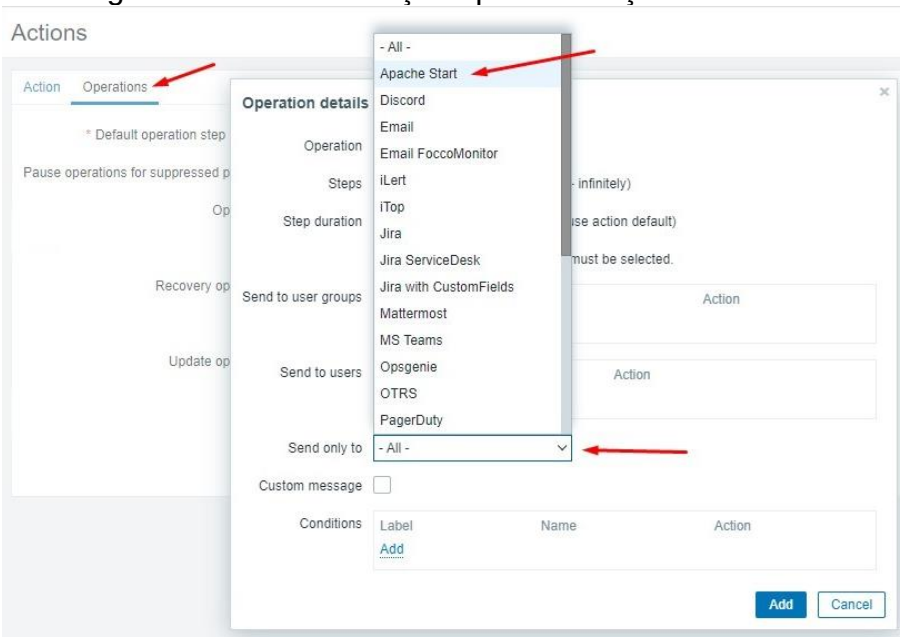
Figura 36 – Setor de *scripts* para correção de alertas



Fonte: Autoria própria.

Logo após, foi necessário configurar uma ação, conforme Figura 37, que possibilitasse o disparo automático do script configurado. Através da seleção do *host*, alerta e comando do script desenvolvido, foi possível que no momento que houvesse a falha referente ao alerta, fosse disparado o script para correção, resultando na correção automática do alerta citado.

Figura 37 – Setor de ações para correção de alertas



Fonte: Autoria própria.

Os casos de testes foram atingidos conforme o planejado. Também se verificou que todos os casos cumpriram as pós-condições estabelecidas, junto com as suas respectivas métricas de monitoramento de servidores (conforme capítulo 2.3). Pode-se confirmar ainda que todos os critérios de validações referentes a ferramentas (conforme capítulo 4) foram atendidas, sendo compatíveis com o objetivo da solução do desenvolvimento estabelecida neste presente trabalho.

Os resultados coletados tiveram como base o recurso de suporte ao profissional de tecnologia da informação responsável por administrar sistemas, visando a garantia de coleta de ocorrências relacionadas a incidentes e situações-problema em tempo real, e agir de acordo conforme a severidade das notificações.

Durante a sondagem de métricas a serem utilizadas no monitoramento, levou-se em consideração da importância das mesmas quando o assunto visa o correto funcionamento da estrutura, além de fácil interpretação e informações importantes que gerem uma reação rápida do profissional que estiver atuando no monitoramento.

7 CONSIDERAÇÕES FINAIS

Com a exigência progressiva na qualidade de monitoramento de aplicações, se fez necessário o surgimento de novas alternativas que pudessem garantir a disponibilidade e a qualidade no fornecimento de entrega de dados e escalabilidade, conforme a aplicação. Para suprir essa demanda, uma vasta gama de opções referentes a ferramentas de monitoramento emergiu, com ênfase para as plataformas que contenham recursos de coleta, provisionamento e entrega de dados em tempo real.

Visto que tais soluções não são acessíveis para grande parte das pessoas e organizações devido ao seu custo elevado, buscou-se soluções que atendessem as necessidades de monitoramento capazes de garantir princípios básicos de infraestrutura, tais como eficiência no monitoramento e alta disponibilidade.

Ao longo deste trabalho analisou-se como a fundamentação teórica de ambientes de monitoramento se relaciona com ferramentas capazes de proporcionar a coleta de dados em tempo real, bem como métricas de baixo nível e a importância fundamental no gerenciamento de recursos computacionais.

O desenvolvimento aconteceu conforme o previsto. Ao longo deste trabalho, o ambiente que viria a hospedar as ferramentas foi preparado e homologado para que pudesse ser feita a instalação, parametrização e comunicação acessível entre todas as plataformas envolvidas. Da mesma forma que houve a coleta de métricas, os casos de teste foram executados com o objetivo de cumprir todas as funcionalidades propostas no planejamento inicial.

No decorrer da validação, as ferramentas envolvidas cumpriram a questão norteadora desta pesquisa (“O que é necessário para um ambiente de monitoramento fornecer alertas em tempo real para um *smartphone* via Telegram?”), além disso, pode-se observar que as funcionalidades da solução proposta mostram como é possível disparar os alertas para monitoramento de sistemas e garantir a entrega da coleta de dados para o usuário final.

REFERÊNCIAS

BOTH, David. **opensource**. An introduction to swap space on Linux systems, 2020.Disponível em: https://opensource.com/article/18/9/swap-space-linux-systems. Acesso em: 15 mai. 2021.

BURROUGHS, Adam. **sbnonline**. Linux servers are becoming more popular for businesses. Here's why, 2017 Disponível em: https://www.sbnonline.com/article/linux-servers-becoming-popular-businesses-heres/. Acesso em: 16 mai. 2021.

CAMPION, Nick. **metricfire**. 9 Best Open Source Network Monitoring Tools, 2021. Disponível em: https://www.metricfire.com/blog/9-best-open-source-network-monitoring-tools/. Acesso em: 16 mai. 2021.

CISCO. **cisco**. What is network monitoring, 2021. Disponível em: https://www.cisco.com/c/pt_br/solutions/automation/what-is-network-monitoring.html. Acesso em: 17 mai. 2021.

DELL. **Dell, Notícias**. Monitoramento de infraestrutura de TI: o poder de prever e prevenir. Brasil, 2021. Disponível em: https://www.service.com.br/monitoramento-de-infraestrutura-de-ti-o-poder-de-prever-e-prevenir/. Acesso em: 29 abr. 2021.

ELLINGWOOD, Justin. **digitalocean**. An Introduction to Metrics, Monitoring, and Alerting, 2021. Disponível em: https://www.digitalocean.com/community/tutorials/an-introduction-to-metrics-monitoring-and-alerting. Acesso em: 15 abr. 2021.

GAIDARGI, Juliana. **infonova**. Os tipos de monitoramento e como funcionam. Brasil, 2021. Disponível em: https://www.infonova.com.br/artigo/tipos-de-monitoramento/. Acesso em: 11 mai. 2021.

GOMES, Pedro. **opservices**. CONHEÇA OS PRINCIPAIS PROTOCOLOS DE REDE E SEUS USOS. Brasil, 2021. Disponível em: https://www.opservices.com.br/protocolos-de-rede/. Acesso em: 20 mai. 2021.

GOMES, Pedro. **opservices**. O QUE É E COMO FUNCIONA O GRAFANA? ENTENDA AQUI! Brasil, 2021. Disponível em: https://www.opservices.com.br/grafana/. Acesso em: 28 mai. 2021.

GRAFANA. **Grafana**. Grafana requirements, 2021. Disponível em: https://grafana.com/docs/grafana/latest/installation/requirements/. Acesso em: 28 mai. 2021.

HAYNES, Derek. **scoutapm**. Understanding Linux CPU stats, 2021. Disponível em: https://scoutapm.com/blog/understanding-linuxs-cpu-stats. Acesso em: 15 mai. 2021.

HEIN, Daniel. **solutionsreview**. The 7 Key Features for Network Monitoring Solutions, 2020. Disponível em: https://solutionsreview.com/network-monitoring/the-7-key-features-for-network-monitoring-solutions/. Acesso em: 10 mai. 2021.

HOLTZ, Matt. **liquidweb**. A Guide to Email Protocols: SMTP, POP3, and IMAP, 2021. Disponível em: https://www.liquidweb.com/kb/a-beginners-guide-to-email-protocols-smtp-pop3-and-imap/. Acesso em: 10 mai. 2021.

HOPE, **computerhope**. Disk capacity, 2020. Disponível em: https://www.computerhope.com/jargon/d/diskcapa.htm. Acesso em: 15 mai. 2021.

INGALLS, Sam. **serverwatch**. What Is a Database Server & What Is It Used For, 2021. Disponível em: https://www.serverwatch.com/guides/database-server/. Acesso em: 22 mai. 2021.

INTEGRAÇÃO ZABBIX. **Zabbix**. Available Solutions, 2021.Disponível em: https://www.zabbix.com/integrations/linux. Acesso em: 22 mai. 2021.

IONOS. **Ionos**. High CPU Usage, what does this mean, 2021. Disponível em: https://www.ionos.com/digitalguide/server/know-how/cpu-usage/. Acesso em: 14 mai. 2021.

KING, Kendall. **velocityhost**. Open Source: Good for business, 2021. Disponível em: https://velocityhost.com.au/blog/open-source-good-for-business/. Acesso em: 14 mai. 2021.

KOROLOV, Maria. **csoonline**. Open source software security challenges persist, 2021. Disponível em: https://www.csoonline.com/article/3157377/open-source-software-security-challenges-persist.html. Acesso em: 14 mai. 2021.

KREEFTMEIJER, Jeff. **appsignal**. Understanding CPU statistics, 2018. Disponível em: https://blog.appsignal.com/2018/03/06/understanding-cpu-statistics.html. Acesso em: 14 mai. 2021.

KUMAR, Ravi. **quora**. Why do some of the companies prefer Linux OS over Windows, 2015. Disponível em: https://www.quora.com/Why-do-some-of-the-companies-prefer-Linux-OS-over-Windows. Acesso em: 12 mai. 2021.

LINUXIZE. **linuxize**. How to Check Memory Usage in Linux, 2020. Disponível em: https://linuxize.com/post/check-memory-linux/. Acesso em: 12 mai. 2021.

LUIZ, WASHINGTON. **CNN**. Telegram supera WhatsApp e é o aplicativo mais baixado do mundo em janeiro. Brasil, 2021. Disponível em: https://www.cnnbrasil.com.br/business/telegram-supera-whatsapp-e-lidera-ranking-de-aplicativos-mais-baixados-em-janeir/. Acesso em: 29 mai. 2021.

MELNICK, Jeff. **netwrix**. Top Best Network Monitoring Tools of 2021, 2021. Disponível em: https://blog.netwrix.com/2021/03/31/network_monitoring_tools/. Acesso em: 16 mai. 2021.

MARIADB. **mariadb**. About MariaDB Server, 2021. Disponível em: https://mariadb.org/about/. Acesso em: 10 nov. 2021.

MARIADB. **mariadb**. Reserved Words, 2021. Disponível em: https://mariadb.com/kb/en/reserved-words/. Acesso em: 10 nov. 2021.

MITCHELL, Bradley. **lifewire**. The internet wouldn't exist without servers, 2021. Disponível em: https://www.lifewire.com/servers-in-computer-networking-817380. Acesso em: 11 mai. 2021.

MOHR, Rodrigo. **UFRGS**. Análise de ferramentas de monitoração de código aberto, 2021. p. 27~34, 2013. Disponível em: https://lume.ufrgs.br/handle/10183/66105. Acesso em: 11 mai. 2021.

NAGIOS. **nagios**. What Nagios Provides, 2021. Disponível em: https://www.nagios.org/about/overview/. Acesso em: 8 nov. 2021.

NASCIMENTO DAS NEVES, ANDRE. **nucleodoconhecimento**. Monitoramento de redes com software livre: Solução de baixo custo em uma escola de rede privada UFPA, 2019. Disponível em: https://www.nucleodoconhecimento.com.br/engenharia-da-computacao/monitoramento-de-redes. Acesso em: 8 nov. 2021.

NASH, Kim. **cio**. Network Monitoring Definition and Solutions, 2007. Disponível em: https://www.cio.com/article/2438133/network-monitoring-definition-and-solutions.html. Acesso em: 21 mai. 2021.

NAYAK, Siben. **freecodecamp**. How to Use Metrics to Monitor Your Microservices, 2021. Disponível em: https://www.freecodecamp.org/news/microservice-observability-metrics/. Acesso em: 21 mai. 2021.

NICHOLS, Steven. **linode**. Linux System Monitoring Fundamentals, 2021. Disponível em: https://www.linode.com/docs/guides/linux-system-monitoring-fundamentals/. Acesso em: 22 abr. 2021.

ORACLE. **Oracle**. Database Concepts, 2021. Disponível em: https://docs.oracle.com/cd/B19306_01/server.102/b14220/intro.htm. Acesso em: 11 abr. 2021.

ORACLE. **Oracle**. Installing Oracle Database, 2021. Disponível em: https://docs.oracle.com/cd/E11882_01/install.112/e24326/toc.htm#BABFCHBA. Acesso em: 11 abr. 2021.

PAL, Kaushik. **techopedia**. What makes application performance monitoring important, 2021. Disponível em: https://www.techopedia.com/what-makes-application-performance-monitoring-important/7/32193. Acesso em: 10 nov. 2021.

PETERSON, Stacey. **searchstorage**. RAM (Random Access Memory), 2021. Disponível em: https://searchstorage.techtarget.com/definition/RAM-random-access-memory. Acesso em: 10 nov. 2021.

POSEY, Brien. **techtarget**. Definition: What is a Server, 2021. Disponível em: https://whatis.techtarget.com/definition/server. Acesso em: 28 mai. 2021.

RAKOSHITZ, Gregory. **patentscope**. Traffic Monitoring Tool for Bandwidth management. Estados Unidos,1999. Disponível em: https://patentscope.wipo.int/search/en/detail.jsf?docId=WO1999034544. Acesso em: 28 mai. 2021.

SATHYANARAYANAN. **opensourceforu**. Ten reasons why We Should Use Linux, 2020. Disponível em: https://www.opensourceforu.com/2020/03/reasons-to-use-linux/. Acesso em: 18 mai. 2021.

SEMATEXT. **sematext**. The Complete Guide to Metrics, Monitoring and Alerting, 2020. Disponível em: https://sematext.com/blog/monitoring-alerting/#toc-what-are-system-metrics-1. Acesso em: 18 mai. 2021.

SHAMSI, Jawwad. **researchgate**. Principles of Network Monitoring. National University of Computer and Emerging Sciences, Karchi, Pakistan, 2009. Disponível em: https://www.researchgate.net/publication/228831271_Principles_of_Network_Monitoring. Acesso em: 18 mai. 2021.

SHARE ZABBIX. **zabbix**. Integração Zabbix com WhatsApp. Brasil, 2021. Disponível em: https://share.zabbix.com/zabbix-tools-and-utilities/cat-notifications/integracao-zabbix-com-whatsapp. Acesso em: 8 nov. 2021.

SIMS, Gary. **linux**. All about Linux swap space, 2007. Disponível em: https://www.linux.com/news/all-about-linux-swap-space/. Acesso em: 13 mai. 2021.

STERNE, Jonathan. **britannica**. plug-in – software, 2021. Disponível em: https://www.britannica.com/technology/plug-in. Acesso em: 13 mai. 2021.

MUDRAKOLA, Sukesh. **techgenix**. Linux security and growing cyberthreats: Everything you need to know, 2021. Disponível em: https://techgenix.com/linux-security/. Acesso em: 21 mai. 2021.

SUVVARI, Ravi. **linkedin**. Differences between Agent vs. Agentless Monitoring, 2018. Disponível em: https://www.linkedin.com/pulse/differences-between-agent-vs-agentless-monitoring-ravi-suvvari. Acesso em: 16 nov. 2021.

TELEGRAM. **telegram**. Telegram APIs, 2021. Disponível em: https://core.telegram.org/api. Acesso em: 4 mai. 2021.

URLOCKER, Zack. **infoworld**. How open source gains market share, 2021. Disponível em: https://www.infoworld.com/article/2632144/how-open-source-gains-market-share.html. Acesso em: 17 mai. 2021.

VANOVER, RICK. **redmondmag**. Hyper-V Private vs. Internal Virtual Switches: Which to Choose, 2021. Disponível em: https://redmondmag.com/articles/2014/07/24/private-vs-internal-virtual-switches.aspx. Acesso em: 15 mai. 2021.

VEESP. **veesp**. Linux System Monitoring Basics, 2017. Disponível em: https://veesp.com/blog/linux-system-monitoring-basics/. Acesso em: 4 mai. 2021.

VENEZIA, Paul. **infoworld**. How to choose the right Linux distro, 2014. Disponível em: https://www.infoworld.com/article/2687088/how-to-choose-a-linux-server-distribution.html?page=2. Acesso em: 13 mai. 2021.

VLADISHEV, A. **Manual Zabbix**, vs 1.6, 17 ed. 2008. Disponível em: Acesso em: 17 mar. 2021