UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Uma arquitetura de monitoramento de infraestruturas virtuais

Dissertação de Mestrado

Lanay Marques Cardoso

São Cristóvão – Sergipe

2021

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Lanay Marques Cardoso

Uma arquitetura de monitoramento de infraestruturas virtuais

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Prof. Dr. Ricardo José Paiva de Britto Salgueiro
Coorientador(a): Prof. Dr. Rubens de Souza Matos Júnior

São Cristóvão – Sergipe

2021

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS.
Candidato: Lanay Marques Cardoso

Em 30 dias do mês de novembro do ano de dois mil e vinte, com início às 17h00min, realizou-se na Sala virtual meet.google.com/okz-fjgs-xpd. A Sessão Pública de Defesa de Dissertação de Mestrado do candidato **Lanay Marques Cardoso**, que desenvolveu o trabalho intitulado: **"Uma arquitetura de monitoramento de infraestruturas virtuais",** sob a orientação do Prof. Dr. **Ricardo José Paiva De Britto Salgueiro**. A Sessão foi presidida pelo Prof. Dr. **Ricardo José Paiva De Britto Salgueiro** (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Prof. Dr. **Rubens De Souza Matos Junior** (PROCC/UFS) , **Prof. Dr. Edward David Moreno Ordonez** (Procc/UFS)e, em seguida, ao Prof. Dr. **Jean Carlos Teixeira De Araujo** (UFAPE). Após as discussões, a Banca Examinadora reuniu-se e considerou o (a) mestrando *(a)* **APROVADA** *"(aprovado/reprovado)"*. Atendidas as exigências da Instrução Normativa 01/2017/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), Resolução nº 25/2014/CONEPE e da Portaria nº 413 de 27 de maio de 2020 (Banca por videoconferência) que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos",* 30 de novembro de 2020.

Prof. Dr. Ricardo José Paiva De Britto Salgueiro
(PROCC/UFS)
Presidente

Prof. Dr. Edward David Moreno Ordonez
(PROCC/UFS)
Examinador Interno

Prof. Dr. Rubens De Souza Matos Junior
(PROCC/UFS)
Examinador Interno à instituição

Prof. Dr. JEAN CARLOS TEIXEIRA DE ARAUJO
(UFAPE/UFS)
Examinador Externo

Lanay Marques Cardoso
discente

Este trabalho é dedicado a Deus, aos meus pais, amigos e professores.

Agradecimentos

Dedico este trabalho aos meus queridos pais. Se cheguei até aqui, foi graças a eles. Agradeço por todo apoio, amor, animação e pelos ótimos conselhos do meu pai, José Cardoso, e por toda força, companheirismo, amor, cuidado e grande esforço da minha mãe, Silvania Marques, que sempre luta por mim. E, também, as infinitas vezes que eles me ouviram.

Este trabalho foi produzido em meio a grandes mudanças: em plena pandemia, isolamento social e a diversas lutas. Se não fosse pelas pessoas que estão a minha volta, não seria possível concluir. Agradeço a Erick Antunes pelas palavras de apoio, pela disposição em ajudar, por muitas vezes me incentivar, por me ouvir bastante e indiretamente mostrar que tudo é possível. Sou grata aos meus orientadores Ricardo Salgueiro e Rubens Matos, por todos puxões de orelha, apoio, direcionamento e orientação. Tenho gratidão pelo meu amigo Miltinho pela paciência, bondade, descontração, pelos momentos de desabafo e principalmente por me ajudar em vários aspectos. Tenho uma enorme gratidão pelos meus professores Fabio Gomes Rocha e Pablo Marques por todo ensinamento durante estes longos anos antes de chegar a esta etapa, por acreditar que chegaria até aqui. Agradeço aos meus amigos, Rafael Rosa, Jordana Naftali, Álvaro Matias e José Francisco por todas palavras de apoio e pelos momentos de alegria. Agradeço aos meus colegas do ELAN José Andeson, Jonathan Cunha, Wesley Oliveira e Itauan Ferreira. Agradeço a todos os outros professores do DCOMP/UFS, PROCC e a secretaria que contribuíram para minha formação.

Você ganha força, coragem e confiança
através de cada experiência
em que você realmente para
e encara o medo.

(Eleanor Roosevelt)

Resumo

Em um mundo cada vez mais conectado, a computação em nuvem agrega todo tipo de informações. Com o crescente número de aplicações online e a grande perspectiva de aumento dos serviços oferecidos na Internet, é essencial monitorar continuamente a disponibilidade das aplicações e serviços, progressivamente suscetíveis a ataques cibernéticos e outros tipos de eventos causadores de falhas. Os softwares para monitoramento, em sua maioria, são proprietários, caros e podem não ser escaláveis; em contrapartida, os softwares livres possuem um design que necessita de um maior tempo de aprendizado. Além das questões citadas, há uma escassez de funcionalidades para o monitoramento de APIs utilizadas pelos serviços e aplicações na web, cada vez mais importantes com a consolidação das arquiteturas de microsserviços. Como proposta de um modelo de monitoramento leve e escalável, esta dissertação apresenta uma arquitetura adaptável que pode ser aplicada a infraestruturas em minutos, sem intervenções profundas que afetem a integridade do sistema original. Um módulo *plug-in* foi desenvolvido para monitorar APIs REST. Os dados de monitoramento são mostrados em um *dashboard* gráfico que utiliza o *plug-in*, identificando a disponibilidade das aplicações. Os dados obtidos dos status desses *endpoints* são armazenados em um banco de dados de tempo real. A arquitetura de monitoramento foi implementada de maneira que seja adaptada para ambiente em nuvem, tanto pública como privada. Os estudos de casos efetuados demonstraram a aplicabilidade da solução, exibindo uma visão geral de serviços monitorados e permitindo a análise dos dados. A arquitetura de monitoramento tem um modelo distribuído adaptativo, o qual é escalável e integrável com diversas ferramentas, nuvens e aplicações.

Palavras-chave: Computação em nuvem, Monitoramento, API, REST.

Abstract

In an increasingly connected world, cloud computing aggregates all kinds of information. With the growing number of online applications and the great prospect of increasing the services offered on the Internet, it is essential to continuously monitor the availability of services and applications, progressively susceptible to attacks cybernetics and other types of failure-causing events. Most monitoring software is proprietary, expensive and may not be scalable; in contrast, free software has a design that requires a longer learning time. In addition to the issues mentioned, there is a lack of functionality for monitoring APIs used by services and applications on the web, which are increasingly important with the consolidation of microservices architectures. As a proposal for a light and scalable monitoring model, this dissertation presents an adaptable architecture that can be applied to infrastructures in minutes, without deep interventions that affect the integrity of the original system. A plug-infoi module developed to monitor APIsREST. The monitoring data are shown on a graphic chart using the plug-in, identifying the availability of the applications. The data obtained from the status of these endpoints are stored in a real-time database. The monitoring architecture was implemented in a way that is adapted to the cloud environment, both public and private. The case studies carried out demonstrated the applicability of the solution, showing a general view of monitored services and allowing data analysis. The monitoring architecture has an adaptive distributed model, which is scalable and integrable with different tools, clouds and applications

Keywords: Cloud computing, Monitoring, API, REST.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
API	*Application Programming Interface*
AKS	*Azure Kubernetes Service*
AWS	*Amazon Web Services*
CEP	*Complex Event Processing*
CPU	*Central Processing Unit*
CSP	*Cloud Solution Provider*
DCOMP	Departamento de Computação
DDS	*Data Distribution Service*
DPI	*Deep Packet Inspection*
DSL	*Domain Specific Languages*
EC2	*Amazon Elastic Compute Cloud*
EMA	*External Monitoring Architecture*
GKE	*Google Kubernetes Engine*
GPL	*General-Purpose Language*
IaaS	*Infrastructure as a Service*
IMA	*Internal Monitoring Architecture*
IP	*Internet Protocol*
NFV	*Network Function Virtualization*
OGF	*Open Grid Forum*
OCCI	*Open Cloud Computing Interface*
PaaS	*Platform as a Service*
SaaS	*Software as a Service*
SDK	*Software Development Kit*

SLA	*Service Level Agreement*
SOA	*Service-Oriented Architecture*
RSL	Revisão Sistemática da Literatura
VM	*Virtual Machine*
UFS	Universidade Federal de Sergipe

Sumário

1

Introdução

Com o crescente número de dispositivos conectados à Internet, o número de aplicações deve aumentar e, consequentemente, o uso de hospedagem em nuvens. Estima-se que temos mais de 31 bilhões de dispositivos conectados à Internet em 2020 e a projeção demonstra, em números, uma crescente (DAVIS, 2018). Este crescimento de dispositivos conectados tende a ter inúmeras aplicações, que, dispostas em diversos lugares, tornam-se alvos mais visados dos ataques cibernéticos.

Com a disposição crescente de aplicações, o uso da hospedagem em nuvem tende a aumentar. Em se tratando de nuvem, em resumo, descrevemos o fornecimento de serviços de computação, abrangendo servidores, armazenamento, banco de dados, software, rede, e entre outros serviços que comumente são hospedados em *Datacenters*. Estes serviços em nuvem são oferecidos pela Internet. Por este motivo, tratamos o termo como nuvem, referindo-se a algo disposto em outra região e/ou país do mundo. Os termos essenciais tratados nesta dissertação são detalhados no Capítulo 2.

Com a grande demanda por IaaS (Infraestrutura como Serviço), as aplicações e serviços dispostos necessitam de serviços de monitoramento, visando se adaptarem à infraestrutura existente. A gestão dessas aplicações e da segurança da informação é uma necessidade que vem se demonstrando urgente, principalmente em redes corporativas. Como todo o ambiente, os serviços precisam ser monitorados, a fim de garantir a integridade, confiabilidade e disponibilidade do sistema, de acordo com as diretrizes da segurança da informação.

A nuvem engloba diversos conceitos e, por conta desta crescente demanda de serviços entregues por meio da Internet, faz surgir o termo XaaS (Tudo como Serviço). A necessidade de análise constante dos recursos utilizados e da disponibilidade exigem que o monitoramento seja adaptável. A virtualização revolucionou a forma como a computação pode ser oferecida, permitindo que aplicativos e serviços utilizem os mesmos recursos. A capacidade faz com que os recursos sejam provisionados e implantados com agilidade em comparação a infraestruturas

tradicionais, garantindo qualidade de serviço. O hardware e software destas infraestruturas virtuais mudam constantemente para entregar recursos para as exigências do mercado, e, em meio à diversidade e troca de informações, o monitoramento é uma peça fundamental. Como os recursos podem ser adicionados ou removidos rapidamente conforme a necessidade das aplicações dispostas na nuvem, a precisão e o planejamento precisam estar correlacionados e um dos principais motivos pela crescente demanda na nuvem são as vantagens para reduzir significativamente os custos associados à aquisição antecipada de hardware, os quais sempre precisam ser melhorados e/ou trocados (ALCARAZ CALERO; Gutiérrez Aguado, 2015).

A Computação em nuvem ainda quebra barreiras e paradigmas por sua eficiência em reduzir custos, oferecendo recursos e serviços computacionais. A sua utilização flexível de recursos, a rápida escalabilidade e a utilização sob demanda e/ou reservada são alguns destes recursos que dão à computação em nuvem um status privilegiado (AKTAS, 2018a). Os provedores oferecem os recursos de computação que podem ser alugados por consumidores que precisam monitorar a utilização eficiente de hardware, bem como a segurança dos serviços, como dados e aplicações. O consumidor da nuvem necessita realizar monitoramento adequado para sua infraestrutura, portanto é essencial que ele possua ferramentas harmônicas para ambiente em nuvem (FAHAD; AHMED; KAHAR, 2017). Para gerenciar adequadamente os cenários complexos que as infraestruturas virtuais baseadas em nuvem possuem, é necessária a adoção de mecanismos de monitoramento da nuvem e a integração adequada de diversas ferramentas, analisando a escalabilidade de monitoramento em um ambiente pré existente.

O monitoramento de nuvem é atualmente formado tanto por soluções projetadas especificamente para nuvens como, também, por soluções originalmente concebidas para monitoramento geral de objetos. Estas soluções são desenvolvidas com suas respectivas plataformas com diferentes aspectos, nos quais a gestão de visualização de dados e acionamento de alertas são prejudicadas, principalmente para casos em que é preciso lidar com infraestruturas híbridas ou compostas por vários provedores distintos, conhecido como *multi-cloud* (SMIT; SIMMONS; LITOIU, 2013). Para isso, é necessário um sistema que monitore e disponha estas informações. Dependendo do software utilizado para monitoramento e gestão das plataformas, há uma necessidade de conhecimento de diversas aplicações de monitoramento para melhor utilizá-las e servi-las nas infraestruturas virtuais e tradicionais. Por conta dessas características da virtualização, faz-se necessário realizar uma investigação contínua do que está sendo utilizado, trafegado e disponibilizado. Para gerenciar adequadamente os cenários complexos que as infraestruturas virtuais possuem, é necessária a adoção de mecanismos de monitoramento da nuvem, tema principal deste trabalho. Pelo grande interesse em monitoramento de inúmeros dispositivos, aplicações, nuvens e afins, notamos uma crescente ramificação das ferramentas em nuvem, com alguns dados apresentados na seção 3.1.1. Muitos softwares para monitoramento exigem maior tempo em implementação e integração na infraestrutura, causando, consequentemente, aumento na curva de aprendizagem. Este fato não somente ocorre em abordagens proprietárias como também em algumas soluções de sistemas de código aberto. Como este não é o melhor cenário,

principalmente para empresas de pequeno e médio porte, é comum encontrarmos alguns serviços funcionando sem monitoramento, também por questões de desempenho, sobrecarga e integridade de dados.

As soluções específicas para nuvem incluem o (ACCELOPS, 2016), (CLOUDKICK, 2008), (AZURE,) e o (AMAZON. . . ,). Soluções genéricas para ambientes convencionais incluem (MRTG, 1995), (ZABBIX, 2020) e (NAGIOS, 2002). O próprio Nagios não se encaixa bem nas infraestruturas de computação em nuvem, devido ao fato de que ele foi projetado para monitorar a infraestrutura física em vez de infraestruturas virtuais altamente mutáveis. Isso é apontado como um problema em (ALCARAZ CALERO; AGUADO, 2015), em relação à carência de software de gerenciamento que facilite integração com a infraestrutura existente e que forneça recursos de monitoramento a consumidores e provedores. Há uma escassez de literatura adequada das soluções de monitoramento em nuvem que ofereçam uma visão dos recursos e persistência de dados dessas ferramentas, de tal forma que uma solução de interoperabilidade possa ser melhor planejada e também possa evitar o aprisionamento com o fornecedor (TAHERIZADEH et al., 2018). Por esses motivos, apresentamos uma arquitetura de monitoramento adaptável.

Pela mesma visão do crescimento das aplicações utilizadas em dispositivos dispostos na Internet, buscamos uma alternativa para agilidade de monitoramento de aplicações, principalmente para as arquiteturas de microsserviços, as quais as *APIs RESTful* se integram. Este trabalho apresenta um sistema escalável, uma arquitetura com *plug-in* para integração com a maioria das aplicações para monitoramento, armazenando os dados em um contêiner de banco de dados de séries temporais (e.g., InfluxDB (INFLUXDB, 2013)) e exibindo relatórios e informações gráficas em um *dashboard* próprio para tal (e.g., Grafana(GRAFANA, 2014)). Este nível de monitoramento garante a observação da disponibilidade do serviço exposto. Extensivamente, o trabalho inclui outras demonstrações de *plug-in* integrados ao Grafana para monitoramento mais abrangente de outras camadas de uma infraestrutura.

A arquitetura proposta permite uma visão geral de serviços monitorados com mecanismos de monitoramento distribuído adaptativo, implantada na infraestrutura e dispensando a curva de aprendizagem que é necessária na maioria dos sistemas de código aberto. Como forma de abranger o monitoramento, o trabalho inclui o *plug-in* para monitorar aplicações *API RESTful*. Esta arquitetura permite uma visão geral de serviços monitorados, que será feito por meio de mecanismos de monitoramento distribuída adaptativa, implantada na infraestrutura.

Essa mesma arquitetura pode ser implementada em uma infraestrutura existente: pode ser como exemplo mostrado na seção de experimentos deste trabalho em uma nuvem pública, disponibilizando o serviço em um contêiner com o mínimo de uso dos recursos, e também pode ser implementada em um *cluster* já existente, apenas adicionando um novo *pod* com o serviço implementado. Neste documento, apresentamos uma arquitetura que checa *endpoints* de *APIs* por meio do serviço *RESTful* para acessar e monitorar aplicações remotas. Em nossa proposta, descrita na Seção 5, há funcionalidades de *health check* de *endpoints* de aplicações com padrão

de serviço *RESTful*, demonstrando-se como um bom exemplo de tecnologia alternativa para gerenciamento e monitoramento. Implementar em infraestruturas não virtuais implicará em algumas desvantagens, como a não escalabilidade gradual e a grande utilização de hardware, demandando aumento nos custos para compra de hardware. Porém, dependendo do cenário, é possível inserir a arquitetura em uma infraestrutura tradicional.

1.1 Problemática

O monitoramento de recursos é um dos principais desafios em ambientes em nuvens, principalmente em monitoramento de nuvens híbridas, onde há uma complexidade maior. Ainda, a possibilidade de *multicloud* aumenta as possibilidades de problemas para tal. Para realizar um monitoramento eficaz dos recursos, precisa-se adaptar o monitoramento ao nível dos objetos monitorados, buscando uma maior imparcialidade dos aplicativos de monitoramento para integrá-los em uma arquitetura (NATU et al., 2016).

A crescente variação de provedores de nuvens públicas traz tanto alguns desafios como também algumas soluções para o desenvolvimento crescente desse ambiente (ALCARAZ CALERO; AGUADO, 2015). Um dos problemas crescentes é o monitoramento para aplicações, principalmente com o aumento gradual do uso de arquiteturas de microsserviços que , em sua maioria, possuem padrões REST. A falta de informações sobre o estado dessas aplicações é um grande desafio. Essas deficiências de soluções para ambientes virtualizados, aplicações e adjacências é uma grande preocupação. Uma das soluções é a utilização de uma arquitetura eficiente em que seja possível uma rápida inserção em uma infraestrutura existente com menor curva de aprendizagem, sendo que a sua escalabilidade seja exponencial e que possua capacidade de integração com outros sistemas, além do monitoramento adequado para aplicações com padrões REST e a possibilidade de expandir o monitoramento para outros padrões.

1.2 Hipótese

É possível oferecer uma arquitetura de monitoramento com viabilidade para monitoramento dos recursos de infraestruturas virtuais heterogêneas, escalável e com capacidade de integração com sistemas terceiros. Deve permitir a inclusão de um sistema de monitoramento em uma infraestrutura existente, considerando a utilização de infraestrutura tradicional (On-Premises). O sistema deve atender ao monitoramento adequando para aplicações com padrões REST.

1.3 Objetivos

De acordo com a questão apresentada, este trabalho tem, como objetivo geral, propor uma arquitetura de monitoramento escalável e integrável com diversas ferramentas, nuvens e

aplicações. O sistema deve atender ao monitoramento de aplicações com padrões REST.

Para atingir tal objetivo, foram estabelecidos os seguintes desígnios específicos:

- Desenvolver um *plug-in* para monitoramento de aplicações com padrões REST;
- Propor uma arquitetura integrável com ferramentas, outras nuvens e aplicações diversas;
- Avaliar o desempenho para escalar o uso do monitoramento de aplicações padrão REST.

1.4 Classificação da pesquisa

Uma pesquisa pode ser classificada de acordo com sua natureza, seus objetivos, sua abordagem e seus procedimentos técnicos. No tocante a sua natureza, esta pesquisa caracteriza-se como pesquisa aplicada, visto que tem como objetivo resolver o problema de descentralização da base de dados de monitoramento. Uma pesquisa aplicada é aquela que tem como objetivo a geração de conhecimentos que podem ser aplicados a situações práticas, com o intuito de solucionar problemas específicos (GERHARDT T.; SILVEIRA, 2009). Quanto a abordagem, esta pesquisa é quantitativa, pois pretende utilizar dados quantificáveis na sua análise, através de estatísticas. De acordo com (FONSECA, 2002), a pesquisa quantitativa descreve as causas de um fenômeno ou as relações entre variáveis através da linguagem matemática. No que diz respeito aos procedimentos técnicos, esta pesquisa classifica-se como bibliográfica e experimental: Bibliográfica porque foi realizada uma busca na literatura por trabalhos relacionados ao tema, a fim de obter um embasamento teórico para responder o problema de pesquisa; Experimental porque o mecanismo será validado através de experimentos, onde o pesquisador irá alocar uma infraestrutura virtual, a fim de observar o comportamento da arquitetura e observar o comportamento das ferramentas utilizadas em conjunto com o *plug-in* oferecido. Para melhor visualização dos métodos que são utilizados, elaborou-se a Tabela 1, com a lógica que reúne as etapas que são adotadas durante o processo de pesquisa.

Tabela 1 – Resumo da classificação da pesquisas.

Quanto à/ao	**Classificação**
Natureza	Aplicada
Objetivos	Exploratória
Abordagem	Quantitativa e qualitativa
Procedimentos técnicos	Pesquisa bibliográfica e experimental

Fonte: Próprio Autor

1.5 Organização do trabalho

Este Capítulo apresentou a problemática que motiva esta dissertação, a hipótese e os objetivos. O restante do documento está estruturado em Capítulos, da seguinte forma: no Capítulo

2 apresenta–se os fundamentos teóricos, para o melhor entendimento do trabalho, tais como: Computação em nuvem, REST API e Virtualização por contêineres. No Capítulo 3 é apresentada uma revisão sistemática e no capítulo 4 os trabalhos relacionados ao contexto desta dissertação. No Capítulo 5 é dedicado à arquitetura de monitoramento e aos resultados obtidos por meio dos experimentos realizados. Por fim, no Capítulo 6 são apresentadas as considerações finais, incluindo as principais contribuições alcançadas, as dificuldades encontradas e possíveis trabalhos futuros.

2

Fundamentação teórica

No presente capítulo é apresentado fundamentos na seção 2.1 sobre Computação em nuvem e suas características essenciais, modelos de serviços e modelos de implantação, os principais conceitos para uma visão sobre os padrões da nuvem. Na seção 2.2 são abordados detalhes sobre a arquitetura de REST API é um abreviação para *REpresentational State Transfer*. Este termo foi apresentado pela primeira vez por Roy Fielding (FIELDING, 2000), como um estilo arquitetônico para sistemas hipermídia distribuídos. Na seção 2.3 trata de uns conceitos mais utilizados na atualidade, uso de *containers* para distribuição das aplicações, facilitando assim arquiteturas do tipo microsserviços. Como também descreve sistemas de orquestração, que facilitar o gerenciamento dos contêineres.

2.1 Computação em Nuvem

A computação em nuvem é amplamente utilizada no meio acadêmico e corporativo, atualmente. Embora o tema ainda seja jovem, acredita-se que foi utilizado pela primeira vez em 2006, em uma conferência de "Estratégias de mecanismo de pesquisa"[1] pelo CEO (Chief Executive Officer) do Google, Eric Schmidt, referindo-se aos dados da Google[2] estarem "em uma nuvem em algum lugar" e ao fato da computação estar empregando os recursos da Internet.

A computação em nuvem é considerada um estilo de computação no qual os recursos são fornecidos aos consumidores através da Internet. O termo nuvem refere-se a qualquer lugar da internet. Como um modelo, a computação em nuvem permite o acesso de qualquer lugar da internet, favorável e sob demanda de recursos compartilhados configuráveis (por exemplo: servidores, armazenamento, bancos de dados, rede, software, análise e inteligência) que podem ser rapidamente provisionados e cedidos com esforço básico, com um gerenciamento ou interação simples do provedor de serviços. Em resumo, a computação em nuvem é um solução de recursos

[1] https://www.google.com/press/podium/ses2006.html
[2] https://www.google.com

compartilhados que são fornecidos aos usuários sem necessidade de uma infraestrutura presente. O modelo de nuvem é composto por cinco características essenciais, três modelos de serviço e quatro modelos de implantação, de acordo com (Peter Mell (NIST), 2006).

2.1.1 Características essenciais

- Autoatendimento sob demanda (*On-demand self-service*): Capacidade de requerer ou dispensar recursos, obtendo acesso direto e sob demanda sem necessidade de interação humana entre o consumidor e o provedor;
- Acesso via rede (*Broad network access*): Os recursos estão disponíveis na rede e seu amplo acesso aos recursos oferecidos pela nuvem garantem o padrão de alta disponibilidade;
- Compartilhamento de recursos (*Resource pooling*); Os recursos oferecidos pelo provedor são agrupados para atender a vários consumidores usando um modelo multi locatário, com diferentes recursos físicos e virtuais atribuídos e reatribuídos dinamicamente, sendo que os consumidores não possuem conhecimento acerca da localização destes recursos. Existe um conhecimento abstrato da localização, como por país, região ou *datacenter*. Exemplos de recursos incluem armazenamento, processamento, memória, dentre outros;
- Elasticidade Rápida (*Rapid elasticity*): Rápida alocação e capacidade elástica para provisionamentos, agilidade para liberação dos mesmos, e, em alguns casos, automaticamente. A escalabilidade é de acordo com a demanda. Os recursos disponíveis tendem a parecer ilimitados, podendo, a qualquer momento, serem apropriados;
- Serviço Mensurável (*Measured Service*): Controle e otimização do uso dos recursos, mensurando exatamente quais recursos da nuvem foram utilizados. Esta medição do uso de recursos proporciona transparência tanto para o provedor quanto para o consumidor do serviço utilizado, o qual pagará somente pelo que foi utilizado. Este conceito contribuiu para o surgimento do modelo *pay-as-you-go*.

2.1.2 Modelos de serviços

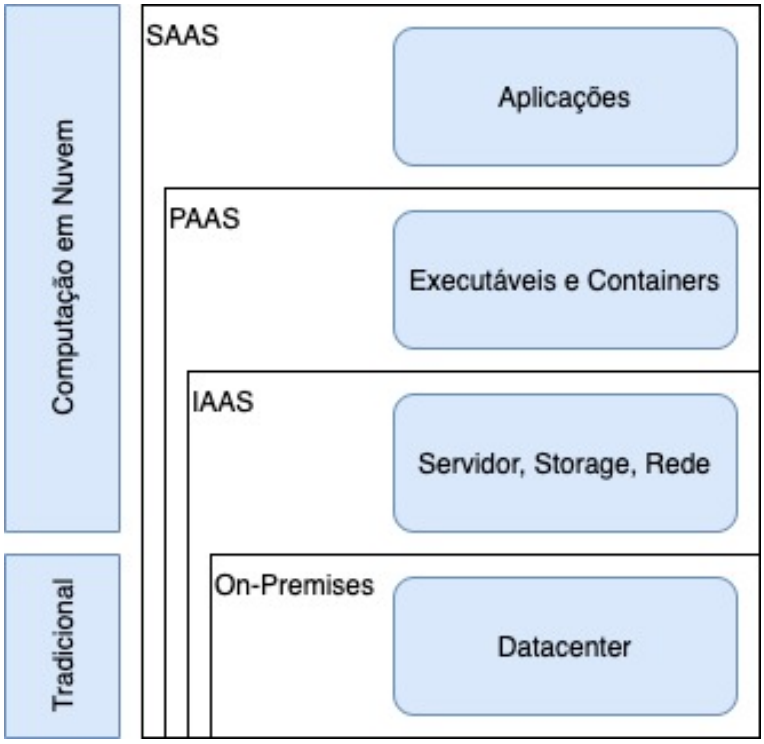
Como base original de modelos de serviços, surgiram os principais, segundo o NIST (*National Institute of Standards and Technology*), detalhamos na Figura 1. Abaixo descrevemos os serviços:

- SaaS - Software como Serviço (*Software as a Service*): um aplicativo que é mantido e gerenciado pelo fornecedor e, por meio da Internet, os clientes controlam e limitam as configurações de software;

- PaaS - Plataforma como Serviço (*Platform as a Service*): Fornece a estrutura para implantar os aplicativos específicos do cliente, que, neste caso, precisam ser compatíveis com a plataforma do provedor;
- IaaS - Infraestrutura como Serviço (*Infraestructure as a Service*): oferece os recursos básicos de computação (armazenamento, conectividade de rede, processamento, etc). Os recursos podem ser virtuais ou físicos, com diferentes níveis de isolamento entre os clientes. Uma grande vantagem é a escalabilidade, que é um dos requisitos essenciais da nuvem.

Depois de vários anos, muitas pesquisas trouxeram outros modelos para definir qualquer coisa como serviço. Como forma de englobar estas discussões, surgiu o termo XaaS - Tudo como um serviço (*Everything as a Service*). O trabalho realizado por (DUAN et al., 2015) mostra a tendência dos serviços pelos diferentes modelos como serviço (aaS), refletindo, parcialmente, as tendências de evolução natural dos serviços.

Figura 1 – Modelos de serviços



2.1.3 Modelos de Implantação

Para alocar os serviços na nuvem, podemos qualificar quatro categorias de modelo, que incluem nuvens privadas, públicas, comunitárias e híbridas.

2.1.3.1 Nuvem pública

Uma infraestrutura de nuvem provisionada para uso aberto pelo público geral. O proprietário de nuvens públicas disponibiliza seus recursos de computação para uma variedade

de usuários pertencentes a diferentes organizações. Um dos principais pontos é a segurança e privacidade, pois é uma tarefa desafiadora ao considerar o monitoramento de nuvem pública. Em nuvens públicas, todas as máquinas virtuais de várias organizações compartilham as mesmas máquinas físicas.

A separação entre as organizações é uma premissa importante, da mesma maneira que o provedor não pode invadir as *VMs* dos usuários. Para atender diferentes clientes, uma camada de software (que pode ser o Hypervisor, se for IaaS) isola os dados entre clientes para que não se enxerguem. Como os usuários da nuvem pública não possuem controle sobre o gerenciamento da topologia física das máquinas e/ou serviços que estão alugando(**??**), sofrem o risco de possíveis indisponibilidades. Um dos pontos cruciais de serviços em nuvens públicas é a redundância em outras regiões (*Datacenters* diferentes) ou hospedagem em outra nuvem pública desses serviços.

As vantagens são o rápido provisionamento, o custo sob demanda e até custos reduzidos frente à Nuvem Privada.

2.1.3.2 Nuvem privada

As nuvens privadas são de propriedade da organização, enquanto as nuvens públicas possuem os serviços alugados para diversas empresas e usuários com seus recursos sendo compartilhados. Em nuvens privadas, tanto as infraestruturas físicas quanto as virtuais pertencem à mesma organização, podendo ser fora do seu próprio *Data center*. Portanto, o monitoramento da nuvem privada possui aspectos próprios, fazendo com que a segurança e o nível de acesso também sejam gerenciados pela organização.

Embora os custos sejam maiores, normalmente é a preferida para o caso de armazenamento de dados estratégicos. Com a nuvem dentro do *Data Center* da empresa, o tempo de resposta será rápido, garantindo uma baixa latência de rede e maior segurança, pois passará pelo *firewall* de perímetro.

Um dos principais benefícios em nuvem privada é a possibilidade de escalabilidade, integração e otimização em hardware, podendo ser considerado um investimento, já que somente a organização privada irá utilizar (De Chaves; URIARTE; WESTPHALL, 2011). A segurança também é um dos pontos fortes em comparação com a nuvem pública, pois os recursos são acessíveis por interfaces internas e conectados por redes privadas, com proteção de firewall de borda e outro interno.

2.1.3.3 Nuvem Comunitária

Semelhante a nuvem privada, porém com a utilização dividida para um grupo de organizações, a nuvem comunitária compartilha as mesmas preocupações como a missão, os requisitos de segurança, políticas e considerações de conformidade. Pode ser gerenciada por mais de uma organização ou, até mesmo, por um terceiro e pode ser fora do *Data Center.* Este

modelo de nuvem é pouco utilizado, mas possui uma vantagem atrativa economicamente, já que os custos são divididos entre as organizações.

2.1.3.4 Nuvem Híbrida

É composta por duas ou mais nuvens distintas que podem ser infraestruturas de nuvens privadas, comunitárias ou públicas que permanecem como uma única entidade. Porém, a tecnologia precisa ser padronizada ou proprietária que permita dados e aplicações com balanceamento ou portabilidade entre elas para que, em caso de uma nuvem ficar instável, possua este balanceamento de carga entre nuvens. A principal desvantagem em uma nuvem híbrida é o tempo de execução em diversos domínios, pois acaba sendo alto e de difícil monitoramento (AKTAS, 2018a).

2.2 REST API

Uma *Application Programming Interface* (API) é composta por instruções e padrões de programação que fornecem dados e informações relevantes de uma determinada aplicação. API RESTful, nosso foco nesta abordagem, é uma API para serviços web que utiliza protocolo HTTP retornando uma série de padrões de informações que podem ser usados de maneiras situacionais, esboçamos a arquitetura de comunicação de aplicações REST API na Figura 2. Com a evolução dos sistemas distribuídos, há uma grande aderência a esse tipo de formato. Além dele, existem arquiteturas que agregam seu formato, facilitando, assim, a implementação do padrão RESTful como *MicroServices*, DDD (*Domain-Driven Design*), dentre outros.

Um serviço RESTful expõe sua posição, dados e finalidade por meio da abstração de recursos. Cada recurso é identificado por um Uniform Resource Identifier. Seu estado pode ter uma ou mais representações e pode ser tratado com um conjunto limitado e predefinido de métodos.

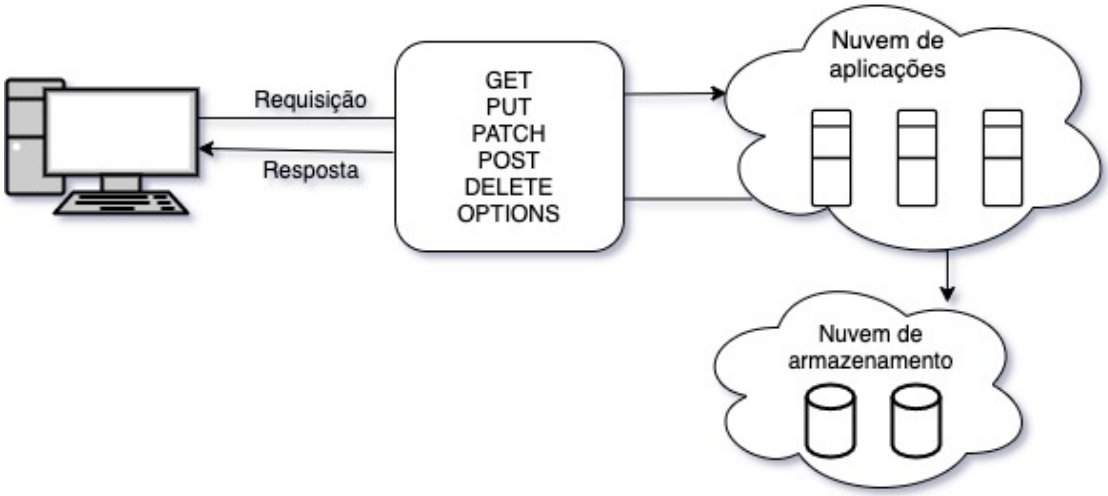
Um recurso pode negociar com clientes qual formato de representação deve ser usado para troca de dados e também pode informar seus clientes sobre os métodos suportados. O conjunto de métodos que podem ser usados para manipular e interagir com um recurso é o seguinte:

- GET: obter o estado atual de uma entidade ou recurso;
- PUT: atualizar o estado de uma entidade ou recurso de maneira idempotente;
- PATCH: modificar um fragmento de uma entidade ou recurso, porém pode não ter contextos idempotentes;
- POST: criar uma entidade ou recurso para o contexto usado;
- DELETE: apagar uma entidade ou recurso;

- OPTIONS: obter quais opções daquele recurso estão disponíveis.

Esses meios são comumente encontrados no protocolo HTTP e carregam semântica adicional que usa interações sem estado. Apenas dois métodos podem não ser idempotentes: POST e PATCH. Enquanto o conjunto de métodos fornecidos por um recurso é fixo aos anteriores, de acordo com o princípio da interface uniforme, o REST não restringe o conjunto de recursos que são publicados por um serviço. Assim, a expressividade da interface reside na seleção de um esquema de nomenclatura de URI e na definição das representações de recursos que são trocadas para cada método de solicitação, em oposição à liberdade de definir um conjunto específico de métodos para cada interface de serviço, como em serviços web tradicionais.

Figura 2 – Arquitetura Rest API



2.3 Virtualização por Contêineres

A conteinerização torna-se cada vez mais prevalente em diversos ambientes. Conhecida, também, como virtualização baseada em *containers*, pode ser considerado uma alternativa leve em relação às máquinas virtuais (VM), pois funcionam em um método de implantação e execução de aplicativos de maneira distribuída sem a necessidade de configurar uma VM completa para cada aplicação. Em vez disso, os contêineres são sistemas isolados, que são executados em um único *host* de controle, acessando um único *kernel*. Esse tipo de arquitetura é vantajosa para microsserviços, onde os aplicativos são distribuídos em várias instâncias, cada uma rodando em um *container* (SOLTESZ et al., 2007).

Consideramos que um contêiner é uma unidade padrão de software, e que nele o código é empacotado com todas as dependências necessárias para que uma aplicação execute. Uma das principais vantagens é a migração deste contêiner de forma rápida e confiável de um ambiente de computação para outro. Pode-se criar uma imagem de contêiner, montando um pacote leve que

inclui tudo que é necessário para executar um aplicativo: código, tempo de execução, ferramentas do sistema, bibliotecas do sistema e configurações, sendo, por isso, um dos pontos de vantagem sobre o uso de contêineres em um ambiente com nuvens, além de ser ideal para uma arquitetura de microsserviços que comumente utiliza o padrão RESTful.

2.3.1 Sistemas de Orquestração

A orquestração de contêineres tem características com base em automatizar a implantação, facilitar o gerenciamento, a escalabilidade e a rede dos contêineres. É ideal para gerenciar e implementar centenas de milhares de contêineres em *hosts*. Esses *hosts* que possuem inúmeros contêineres são chamados de Cluster. Em um Cluster, são alocadas várias máquinas e esse conjunto de máquinas são chamados de nós, que executam aplicativos em contêineres. Cada Cluster deve ter, pelo menos, um nó de trabalho (PAHL et al., 2017).

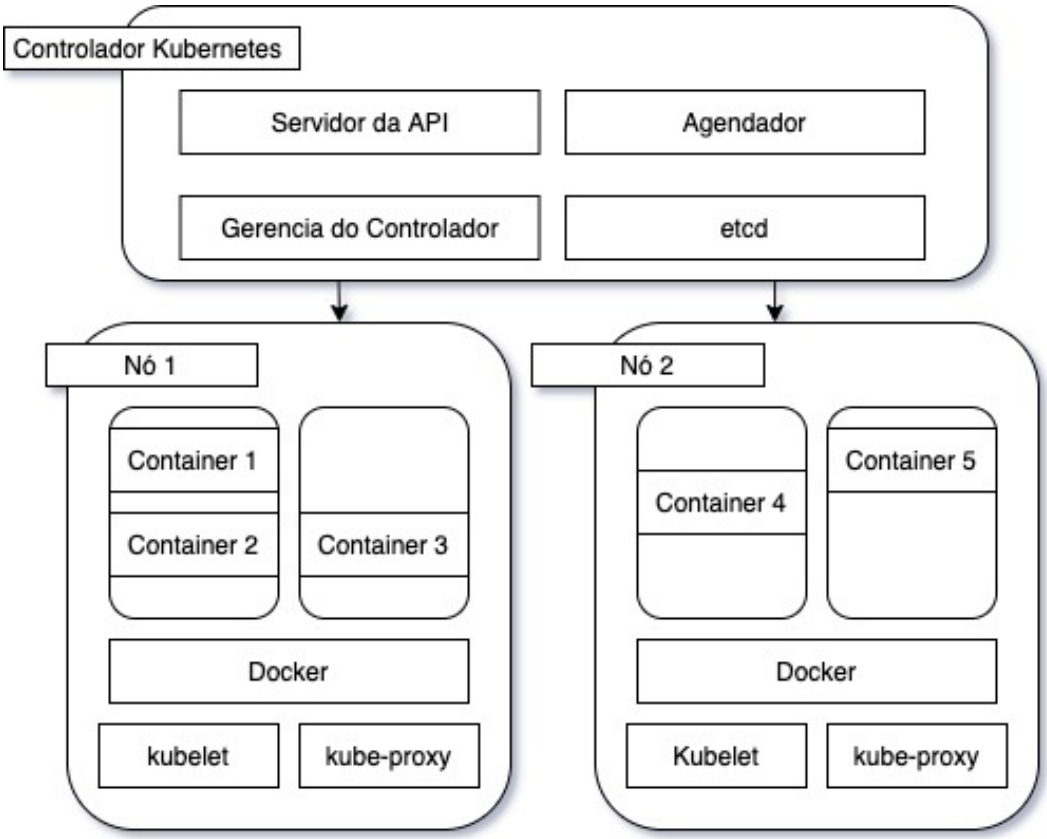
Ao longo dos anos, inúmeras ferramentas e estruturas de orquestração foram desenvolvidas, a fim de simplificar a implantação e execução dos aplicativos na nuvem. São várias as Ferramentas de orquestração de contêineres, sendo que algumas delas estão dispostas em nuvens públicas como Amazon ECS [3] e GKE (*Google Kubernetes Engine*) [4]. Além disso, existem outros aplicativos que podem ser alocados em máquinas ou até mesmo em VMs, como o Kubernetes (KUBERNETES, 2014), Docker Swarm (DOCKER. . . ,), Diego (DIEGO,), Marathon (MARATHON,) e Heat (HEAT,), sendo que todas aplicações trabalham para provisionar ou agendar containers de aplicativos dentro do cluster.

O Kubernetes é um dos principais orquestradores. Em seu sistema, executa as cargas de trabalho colocando os contêineres em *pods* para serem executados em nós. Um nó pode ser uma máquina virtual ou física, dependendo do Cluster. Cada nó contém os serviços necessários para executar *pods*, gerenciados pelo plano de controle.

Na Figura 3, é possível perceber uma explanação sobre a arquitetura do Kubernetes. O ambiente consiste em um sistema mestre, no qual se mantém um registro de todos os objetos, onde, dentro desse contexto, estão os principais componentes: *API Server, Scheduler, Controller-Manager e Etcd*. Estes componentes podem ser executados em um único nó mestre ou serem replicados para uma alta disponibilidade. Os *worker nodes* são máquinas que executam contêineres e são gerenciados pelos nós mestres. O kubelet é o controlador principal e é o responsável por guiar até a camada de execução, geralmente chamada de Docker.

[3] https://aws.amazon.com/ecs/
[4] https://cloud.google.com/kubernetes-engine

Figura 3 – Explanação da arquitetura do Kubernetes



3

Revisão Sistemática

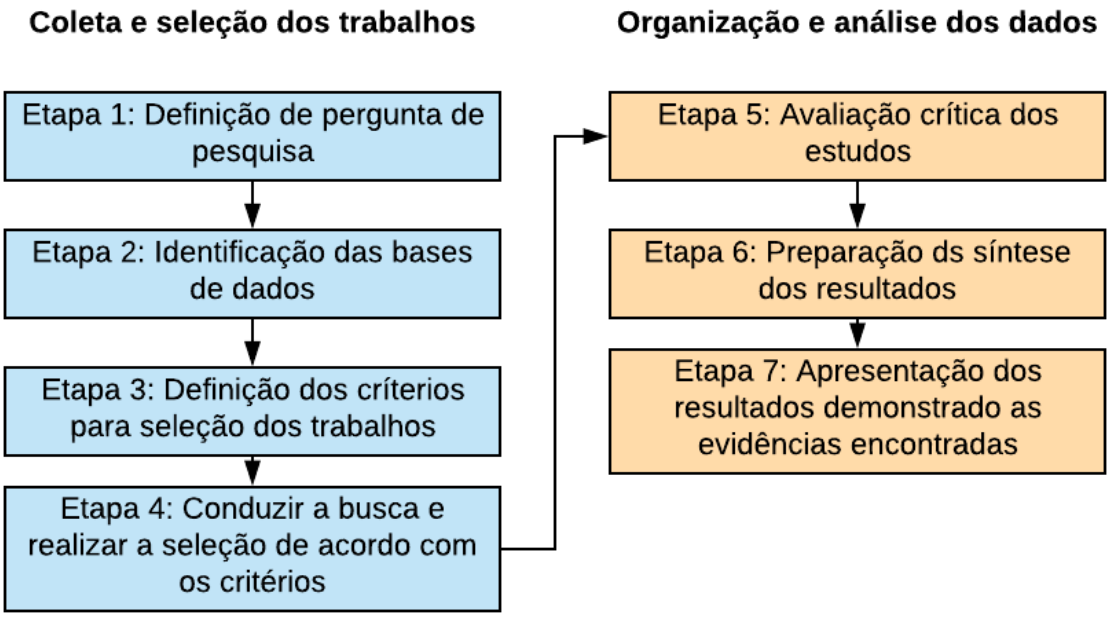
O processo de revisão sistemática da literatura (RSL) difere de uma revisão bibliográfica. Enquanto uma revisão bibliográfica é uma busca aleatória por trabalhos relacionados, na revisão sistemática existe um protocolo de busca bem definido, de forma que outros pesquisadores, ao seguir os passos definidos pela revisão sistemática, devem ser capazes de encontrar os mesmos resultados.

A revisão foi realizada com a aplicação do método de revisão sistemática. De acordo com (PETERSEN, 2008), a revisão sistemática consiste na definição das questões de pesquisa, do escopo de interesse da revisão, da realização de buscas em bases de trabalhos científicos para obtenção dos estudos primários e filtragem dos estudos com base no tema de interesse. De posse dos trabalhos, é necessário selecioná-los e capturar os relevantes, seguindo às questões de pesquisa definidas, e realizar uma leitura cuidadosa a fim de obter as respostas para as questões e apresentar as análises dos resultados obtidos. Por meio de uma revisão sistemática, é possível identificar, avaliar e interpretar todos os trabalhos disponíveis relevantes para uma determinada questão de pesquisa, área de tópico ou fenômeno de interesse. Estudos individuais que contribuem para uma revisão sistemática são chamados de estudos primários.

Há várias razões que justificam a realização de uma revisão sistemática. De forma breve, pode-se dizer que por meio dela é possível resumir evidências sobre determinada área, identificar lacunas para mais investigação e fornecer uma estrutura para novas atividades de pesquisa. Vale ainda ressaltar que a condução da revisão sistemática deve ser realizada de forma justa, seguindo uma estratégia de pesquisa pré-definida, que permita avaliar a integridade da pesquisa e identificar quais apoiam ou não as suas hipóteses (KITCHENHAM, 2004).

Este capitulo descreve como foi realizado os métodos da busca e seleção dos trabalhos, e quais critérios foram utilizados na filtragem dos mesmos.

Figura 4 – Fases de uma revisão sistemática (SAMPAIO R. F.; MANCINI, 2007)



Como observado na Figura 4, a RSL é constituída das fases de planejamento, execução e análise. Na fase de planejamento, devem ser elaboradas as questões de pesquisa para análise dos estudos primários, a seleção das bases onde as buscas serão realizadas e a definição das strings de busca que serão utilizadas. Na fase de execução, as buscas utilizando as strings de busca são realizadas. Além disso, a filtragem dos estudos primários, considerando os critérios de inclusão e exclusão, é feita. Na fase de Análise, os dados são sintetizados para responder as questões de pesquisa e um resumo dos trabalhos selecionados é realizado.

É importante diferenciar mapeamento sistemático de revisão sistemática. Ambos seguem o mesmo protocolo para realização da pesquisa, entretanto as características do mapeamento são mais quantitativas, amplas e geralmente sumarizadas em gráficos, enquanto as características da RSL são mais qualitativas, com análises críticas descritivas, minuciosas e profundas, visando elucidar novas evidências e aspectos relevantes através do resumo dos estudos selecionados. Para a fase inicial deste projeto de dissertação, são apresentados ambos dados do mapeamento e da revisão sistemática.

3.1 Planejamento

Na fase de planejamento, o protocolo da revisão sistemática é criado e são definidos os objetivos do trabalho, as hipóteses de pesquisa, as questões de pesquisa, as bases e strings de buscas que serão utilizadas e os critérios de inclusão e exclusão.

3.1.1 Questões de Pesquisa

O passo inicial foi definir a área de interesse da pesquisa. O planejamento inicial foi para o tópico de arquitetura de monitoramento de infraestruturas virtuais. Com o amadurecimento da pesquisa, foi escolhida a opção de fechar o escopo em monitoramento de ambientes em nuvens, pela questão do tema inicial ser jovem no âmbito acadêmico para início de um mapeamento. Com um tema mais abrangente, a revisão melhorou a visão para o escopo.

Para orientar o estudo, foram definidas algumas questões de pesquisa, tendo em vista que essas questões, ao serem respondidas durante a pesquisa, fortaleceriam o conhecimento teórico e as características experimentais necessárias para validar as soluções. Desse modo, foram elaboradas as seguintes questões de pesquisa:

Tabela 2 – questões de pesquisa e dados a serem extraídos.

	Questões de Pesquisa
QP1 3.4.1	Quais são as ferramentas mais comuns/populares na literatura para realizar monitoramento de infraestruturas virtualizadas, sejam elas públicas ou privadas?
QP2 3.4.1	Existem ferramentas que possuem funcionalidades para monitoramento de infraestruturas híbridas (pública e privada)? Caso existam, quais são elas e quais suas vantagens e desvantagens?
QP3 3.4.1	Quais são os tipos de arquitetura de monitoramento para infraestruturas virtualizadas encontrados na literatura?
QP4 3.4.1	Quais são as principais métricas ou informações monitoradas em infraestruturas computacionais virtualizadas?

3.1.2 Estratégias de Buscas

A partir das questões de pesquisas e de leituras prévias de trabalhos foram definidas as palavras-chave e seus sinônimos para compor as strings de buscas. As strings foram construídas utilizando operadores lógicos AND e OR para aumentar e restringir o escopo das buscas. Foram realizados testes de buscas para calibração das strings. As strings de busca são apresentadas na Tabela 3.

Tabela 3 – Strings de busca.

	Strings
Inglês	("Cloud computing"AND monitoring AND management AND architecture)
Português	((("computação em nuvem") OR "infraestrutura virtual") AND monitoramento AND ("Monitoramento de recursos")

Essa string foi aplicada nas bases escolhidas para a realização de buscas que serão detalhadas no próximo tópico na tabela .

3.1.3 Bases de Busca

As escolhas das bases de dados se deram com base na relevância dessas bases para a pesquisa. Também foi objetivo obter a maior cobertura possível de resultados relacionados. Todas as buscas foram realizadas através da Internet, a partir dos portais de busca disponibilizados pelas bases de trabalhos. Alguns desses portais de busca forneciam ferramentas melhores de busca, por exemplo a possibilidade de implementar filtros mais avançados, enquanto outros não possibilitaram buscas com expressões lógicas. Não foram considerados intervalos de datas especificas, já que o assunto em si é um problema clássico da literatura. As bases escolhidas são apresentadas na Tabela.

Tabela 4 – Bases de busca.

Base	**Site**
Scopus	https://www.scopus.com
IEEE Xplore	https://www.ieee.org/
Elsevier	https://www.sciencedirect.com/
BDBCOMP	http://www.lbd.dcc.ufmg.br/bdbcomp
BDT CAPES	http://bancodeteses.capes.gov.br
Springer Link	https://link.springer.com/
Citeseer library	http://citeseer.ist.psu.edu
Engineering village	https://www.engineeringvillage.com

3.1.4 Critérios de Inclusão e Exclusão

Os critérios de inclusão e exclusão devem servir para classificar os estudos primários e definir quais devem ser considerados ou não nas etapas subsequentes do processo de revisão sistemática. Eles devem ser bem definidos para possibilitar uma classificação confiável e replicável. Com o propósito de analisar quais artigos poderão ser utilizados para responder às questões de pesquisa, foram definidos critérios de inclusão e exclusão, sendo eles:

Critérios de Inclusão:

- Os estudos devem ser trabalhos completos;
- Os estudos devem estar disponíveis na web;
- Os estudos devem ter sidos publicados em journals, simpósios ou conferências das bases citadas;
- Os estudos devem estar relacionados com monitoramento em ambiente de nuvens.
- Os estudos devem estar relacionados com propostas de arquiteturas.

Critérios de Exclusão:

- Estudos duplicados;
- Estudo resumido;
- Estudos que não contêm abordagens sobre arquitetura de monitoramento;
- Estudos que não abordem o monitoramento em ambiente de nuvens como tema ou estudo de caso;
- Estudos que abordam várias arquiteturas de monitoramento em ambiente de nuvens para qualificar.

Após a aplicação desses critérios, 23 artigos se mantiveram na pesquisa. Esses artigos são definidos como estudos relevantes.

3.2 Execução

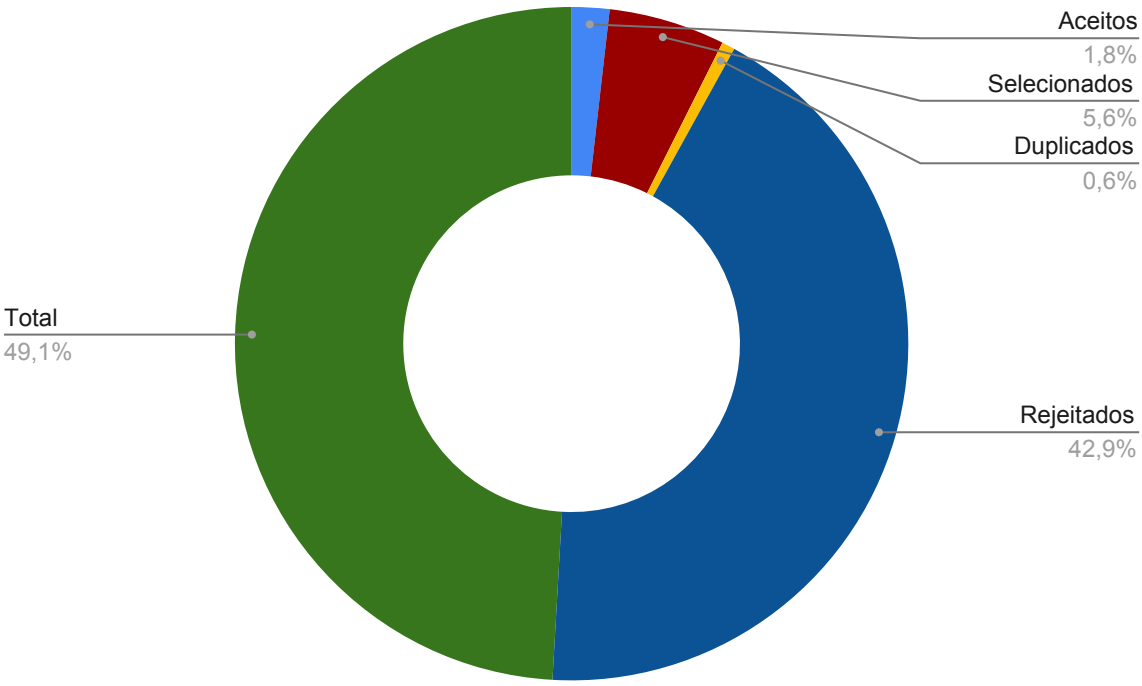
A fase de execução compreende a seleção de estudos e extração dos dados. A definição do protocolo dessa revisão sistemática foi iniciado em dezembro de 2018. Já as fases de busca, leitura e seleção dos trabalhos ocorreram de outubro a dezembro de 2018. As buscas na base IEEE xplore resultaram em 143 trabalhos, na base Science Direct (Elsevier) 61, Scopus resultou em 78 trabalhos, BDBCOMP, base brasileira de trabalhos em computação, resultou em 4 trabalhos, já no Banco de Dissertações e Teses da CAPES (BDT CAPES) foram encontrados 24 trabalhos, na Springer Link foram encontrados 134 trabalhos, na Citeseer Library foram encontrados 32 artigos e na Engineering village foram encontrados 148. Esses resultados são apresentados na Tabela 5.

Tabela 5 – Trabalhos obtidos no estudo primário.

Base	**Estudos primários**
Scopus	78
IEEE Xplore	143
Elsevier	61
BDBCOMP	4
BDT CAPES	24
Springer Link	134
Citeseer library	32
Engineering village	148
Total	624

Conforme as pesquisas realizadas nas bases citadas, foram encontrados 624 estudos sobre o assunto no total. Removendo os 8 duplicados e aplicando os critérios de inclusão, apenas 71 foram selecionados e, utilizando os critérios de exclusão, restaram 23 aceitos, conforme demostração na imagem do gráfico 5:

Figura 5 – Trabalhos aceitos, selecionados, duplicados, rejeitados e totais



3.3 Análise

Na fase de análise a intenção é sintetizar os dados, apresentar e analisar os resultados, a fim de encontrar evidências de que os estudos atendem aos propósitos da revisão sistemática, ou seja, que respondam ao menos uma das questões de pesquisa. Nessa fase, todos os trabalhos que passaram nas fases de inclusão e exclusão foram explorados. Os trabalhos selecionados após essa leitura podem ser vistos na Tabela 6.

ID	Ano	Título	Referência	Base
EP1	2016	Design of a Cloud Monitoring System Beyond Nagios	(CIUFFOLETTI, 2016)	Scopus
EP2	2015	Towards cross-layer monitoring of cloud workflows	(KUBLER; MINOR, 2015)	Scopus
EP3	2017	ConMon: An automated container based network performance monitoring system	\ (MORADI et al., 2017)	Scopus
EP4	2018	CloudProcMon: A non-intrusive cloud monitoring framework	(SYED et al., 2018)	Scopus
EP5	2011	Toward an Architecture for Monitoring Private Clouds	(De Chaves; URIARTE; WESTPHALL, 2011)	IEEE
EP6	2015	MonPaaS: An adaptive monitoring platformas a service for cloud computing infrastructures and services	(ALCARAZ CALERO; AGUADO, 2015)	IEEE
EP7	2014	Truesource: A True Performance For Hierarchical Cloud Monitoring	(Currie A.R., McConnell A., Parr G.P.; K., 2014)	IEEE
EP8	2017	Scalable Agentless Cloud Network Monitoring	(BRATTSTROM; MORREALE, 2017)	Engineering village
EP9	2016	A Distributed Architecture for Monitoring Private Clouds	(PEREZ-ESPINOZA et al., 2016)	Engineering village
EP10	2010	An architecture model of management and monitoring on Cloud services resources	(SUN et al., 2010)	Engineering village
EP11	2015	Um framework para a construção automatizada de cloud monitoring slices baseados em múltiplas soluções de monitoramento	(CARVALHO, 2015)	BDT CAPES
EP12	2013	Uma estratégia de gerenciamento de infraestrutura de datacenters baseada em técnicas de monitoramento distribuído e controle centralizado	(SÁ, 2013)	BDT CAPES
EP13	2014	Uma arquitetura de computação autônoma e cognitiva para monitoramento de nuvens	(SCHUBERT, 2014)	BDT CAPES
EP14	2018	Hybrid cloud computing monitoring software architecture	(AKTAS, 2018b)	Citeseer library
EP15	2016	IaaSMon: Monitoring Architecture for Public Cloud Computing Data Centers	(GUTIERREZ-AGUADO; Alcaraz Calero; Diaz Villanueva, 2016)	Citeseer library
EP16	2013	GMonE: A complete approach to cloud monitoring	(MONTES et al., 2013)	Citeseer library
EP17	2017	Distributed decentralized collaborative monitoring architecture for cloud infrastructures	(XU; CHEN; Alcaraz Calero, 2017)	Citeseer library
EP18	2012	Towards autonomic detection of SLA violations in Cloud infrastructures	(EMEAKAROHA et al., 2012)	Elsevier
EP19	2012	A Self-adaptive hierarchical monitoring mechanism for Clouds	(KATSAROS et al., 2012)	Elsevier
EP20	2016	JTangCMS: An efficient monitoring system for cloud platforms	(LU et al., 2016)	Elsevier
EP21	2013	DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds	(POVEDANO-MOLINA et al., 2013)	Elsevier
EP22	2013	Distributed, application-level monitoring for heterogeneous clouds using stream processing	(SMIT; SIMMONS; LITOIU, 2013)	Elsevier
EP23	2013	Design and implementation of a trusted monitoring framework for cloud platforms	(ZOU et al., 2013)	Elsevier

Tabela 6 – Estudos primários selecionados.

3.4 Resultados da Análise

Aqui são apresentados os resultados quantitativos das bases de buscas, ano das publicações, tema de interesse e questões de pesquisa, e países que foram obtidos através da extração de dados dos 23 artigos que atenderam aos critérios do processo e ficaram na etapa final da revisão sistemática. Após todas as analises, o número de estudos relevantes foram 23, que serão exibidos na Tabela 7.

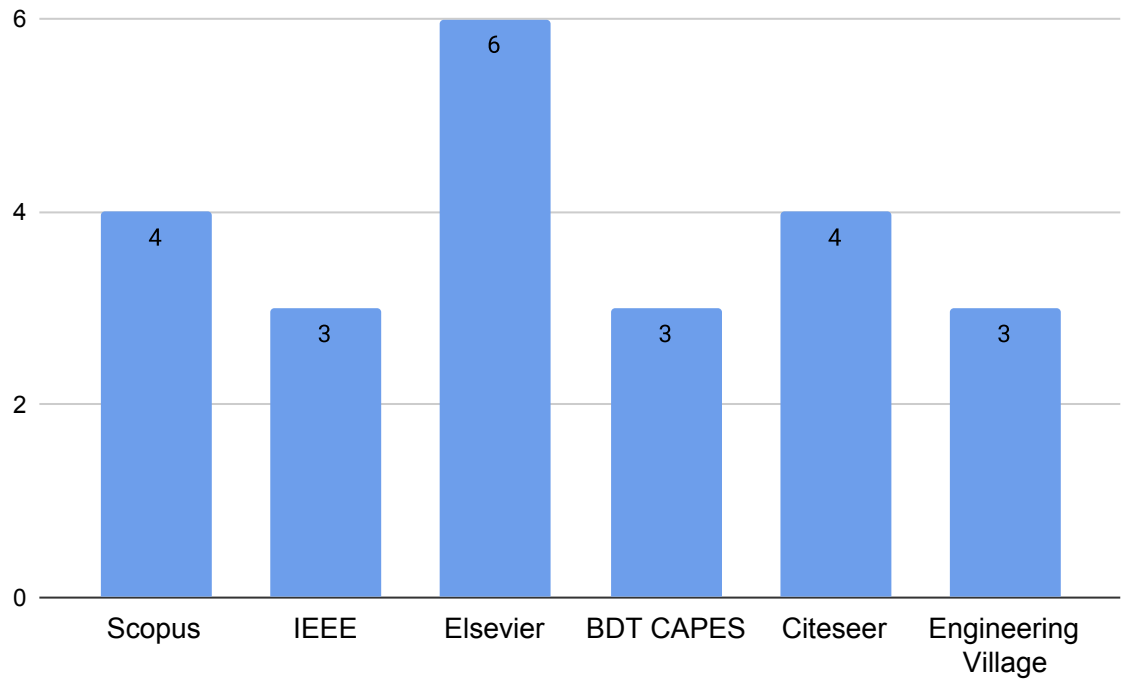
Tabela 7 – Trabalhos selecionados.

Base	Estudos Obtidos
Scopus	4
IEEE Xplore	3
Elsevier	6
BDT CAPES	3
Citeseer library	4
Engineering village	3
Total	23

Base dos Trabalhos

A Tabela 7 e a Figura 6 apresentam o total de estudos selecionados para extração de dados por base de busca. Nota-se que a maioria dos trabalhos são da base Elsevier (6 trabalhos), seguido por Scopus e Citeseer Library com 4 artigos. Com 3 estudos, são as bases IEEE xplore, Engineering village e a BDT Capes.

Figura 6 – Quantidade de trabalhos selecionados por base de busca



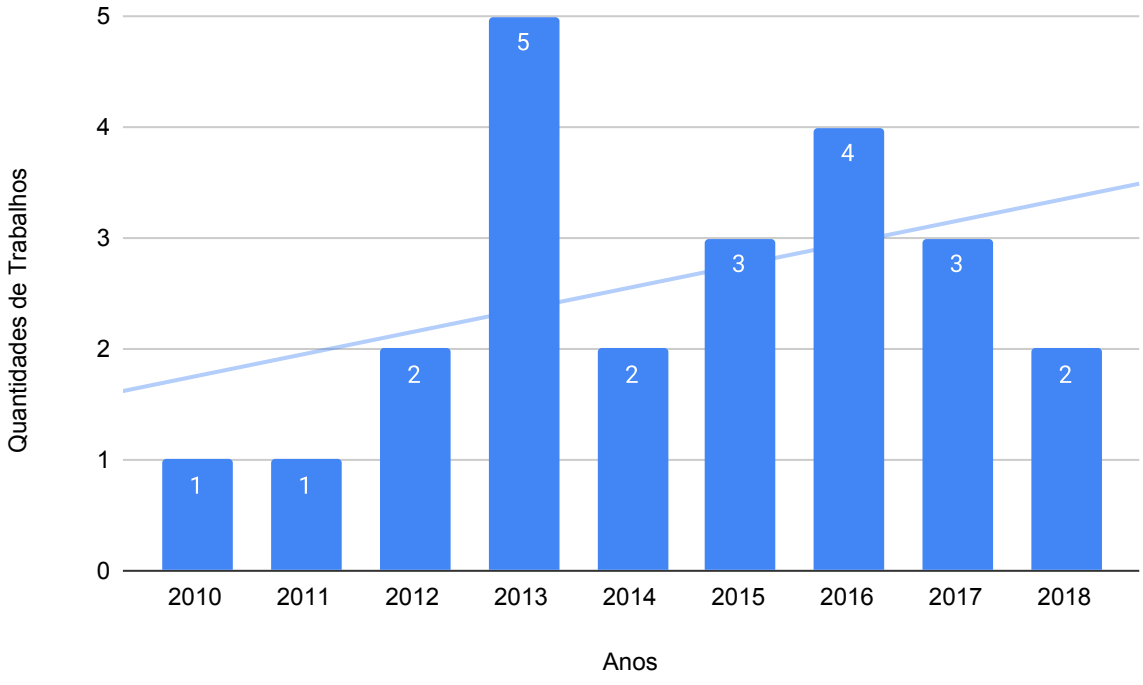
Ano das Publicações

Também foi feito um estudo sobre o ano das publicações, ilustrado na Figura 7. Essa questão de pesquisa busca caracterizar a evolução do uso de ferramentas de monitoramento para ambientes virtuais como nuvens, em especial em nuvens híbridas, a partir da análise do número de estudos relevantes publicados, além dos veículos de publicação dos mesmos.

Observamos que o estudo mais antigo selecionado refere-se ao ano de 2010, e deste ano até 2018 houveram trabalhos selecionados. A maioria dos estudos foram publicados nos anos de 2013 e 2016, com 5 e 4 artigos respectivamente nestes anos, seguidos pelos anos de 2015 e 2017 com 3 trabalhos, depois com os anos de 2012, 2014 e 2018 com 3 pesquisas e, por fim, 2010 e 2011, quando houve apenas 1 trabalho em cada ano. Isso mostra que os temas aqui abordados foram pouquíssimo considerados no período antes de 2010, demonstrando maior interesse entre 2013 a 2018.

Considerando que o tema abordado sobre monitoramento em infraestruturas convencionais é um pouco mais antigo, embora recente em tecnologias virtuais, é possível concluir que estão sendo feitos esforços para atender as necessidades no âmbito de monitoramento em infraestrutura virtual, especificamente para ambientes em nuvens públicas e privadas, justificando o crescimento no interesse em pesquisas que tratam de temas de monitoramento de infraestruturas virtuais. No entanto, a tecnologia ainda é considerada imatura e apresenta-se em fase de crescimento, principalmente em termos de nuvens híbridas, o que oferece inúmeras possibilidades de oportunidades de pesquisa para problemas em aberto.

Figura 7 – Quantidade de trabalhos selecionados por ano

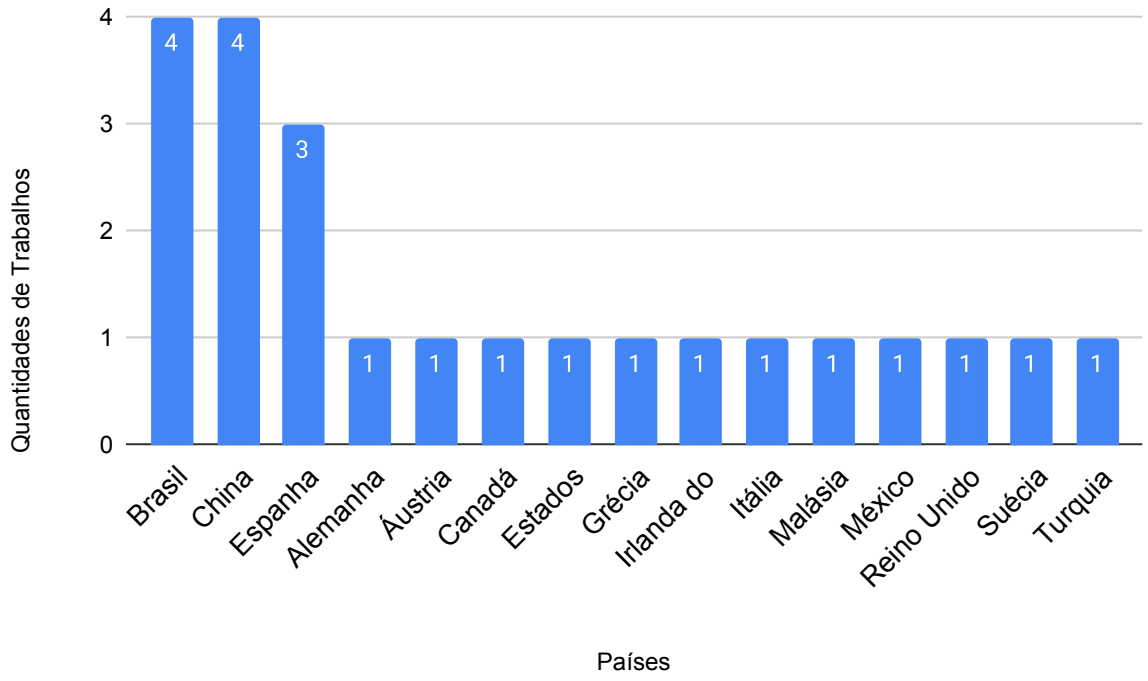


Observa-se que houve uma tendência crescente de trabalhos a partir de 2013, é um tema abrangente que demonstra forte crescimento para os anos seguintes.

Países das Publicações

Outro dado coletado relaciona a quantidade de artigos por país que deu origem ou participou da pesquisa. A Figura 8 apresenta este resultado. A China e o Brasil lideram o ranking com a participação de 4 trabalhos, seguidos pela Espanha com 3 trabalhos. Áustria, Grécia, Irlanda do norte, Reino Unido, Malásia, Alemanha, Suécia, Turquia, México, Itália e Estados Unidos apareceram em 1 artigo cada.

Figura 8 – Quantidade de trabalhos selecionados por país



As pesquisas brasileiras selecionadas foram bastante relevantes. Seguem uma tendência para resoluções de problemas em monitoramento de infraestruturas híbridas, com técnicas para gerenciamento da arquitetura de computação. As quatro selecionadas foram:

Tabela 8 – Trabalhos selecionados no Brasil.

Citação	Título
(CARVALHO, 2015)	Um framework para a construção automatizada de cloud monitoring slices baseados em múltiplas soluções de monitoramento.
(SÁ, 2013)	Uma estratégia de gerenciamento de infraestrutura de datacenters baseada em técnicas de monitoramento distribuído e controle centralizado
(SCHUBERT, 2014)	Uma arquitetura de computação autônoma e cognitiva para monitoramento de nuvens
(De Chaves; URIARTE; WESTPHALL, 2011)	Toward an Architecture for Monitoring Private Clouds

Dentre os projetos explorados, a arquitetura do sistema PCMONS (De Chaves; URIARTE; WESTPHALL, 2011) está sendo amplamente utilizada em outras pesquisas, precisamente citado em 8 artigos dos selecionados na revisão. O PCMONS (Private Cloud Monitoring System) é aplicado somente para nuvens privadas em uma solução de código aberto. Uma das características presentes nesta pesquisa é a integração com o Nagios, e sua abordagem traz um conteúdo relevante para trabalhos futuros.

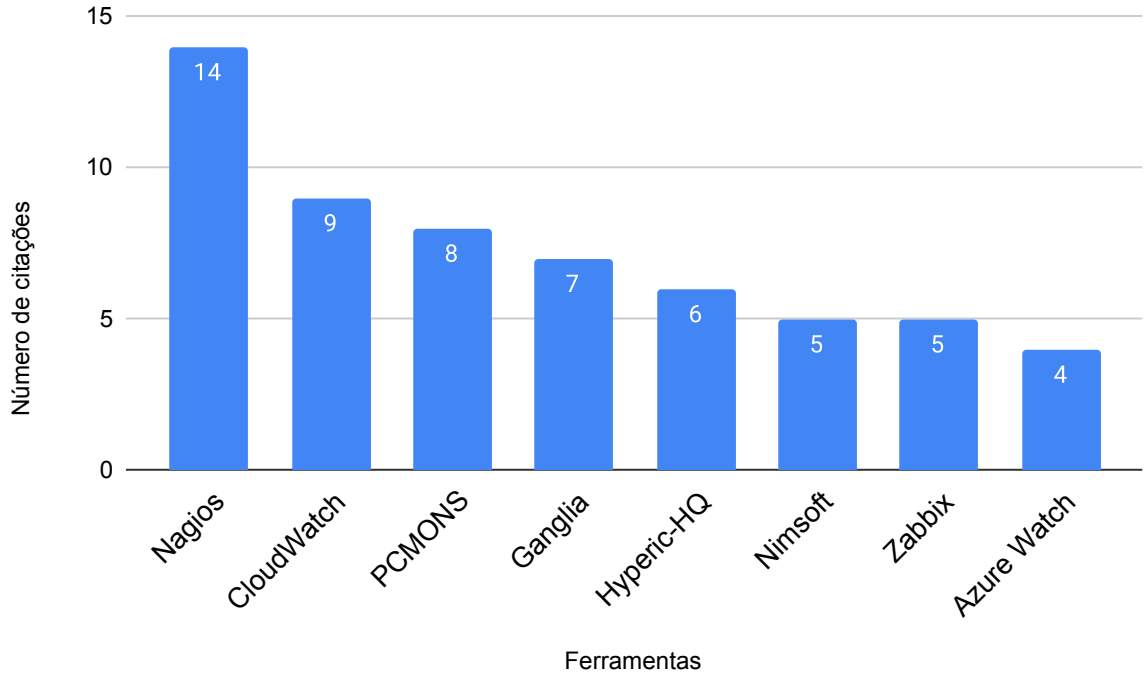
3.4.1 Respostas às Questões de pesquisa

Questão de Pesquisa QP1

Quais são as ferramentas mais comuns/populares na literatura para realizar monitoramento de infraestruturas virtualizadas,sejam elas públicas ou privadas?

Essa questão de pesquisa tem o propósito de apresentar as aplicações populares encontradas para monitoramento no ambiente virtual. Inicialmente são listados os nomes das ferramentas utilizadas nos estudos obtidos a partir da leitura dos mesmos. As ferramentas são listadas na Figura 9, com o quantitativo de artigos que utilizaram as aplicações.

Figura 9 – Ferramentas Populares



Como podemos notar, o Nagios é amplamente utilizado no meio acadêmico e profissional, sendo uma ferramenta com anos de existência. Muitos trabalhos (SMIT; SIMMONS; LITOIU, 2013) selecionam o Nagios devido a sua maturidade e as incríveis possibilidades de adaptação com seus plug-ins e extensões disponíveis, além da facilidade de descobertas automáticas de novas máquinas por meio de *scripts*/executáveis. No artigo (CIUFFOLETTI, 2016), o Nagios é a pedra angular. A ferramenta é considerada bem sucedida e robusta e os objetivos de utilizá-la são pela sua arquitetura de software orientada a plug-ins, pelo controle distribuído das atividades de monitoramento e a adaptação para API que podem controlar a arquitetura, além da atividade da infraestrutura. Outro artigo (ALCARAZ CALERO; AGUADO, 2015) utilizou o Nagios integrado ao OpenStack, que é uma pilha de computação em nuvem também de código aberto. O principal objetivo daquela pesquisa foi fornecer informações sobre a infraestrutura para o provedor de nuvem.

Nesta questão, direcionamos uma investigação detalhada da literatura de monitoramento de nuvem, tanto da indústria quanto da acadêmica. As nuvens públicas conhecidas no mercado geralmente possuem suas próprias ferramentas para monitoramento. Essas ferramentas possuem o objetivo comum de obter os registros dos serviços (rede, servidores, aplicações, entre outros) da infraestrutura que está na nuvem. Desta forma, é possível obter uma melhor compreensão do que é necessário para o ambiente, como aumento ou diminuição dos recursos a serem utilizados.

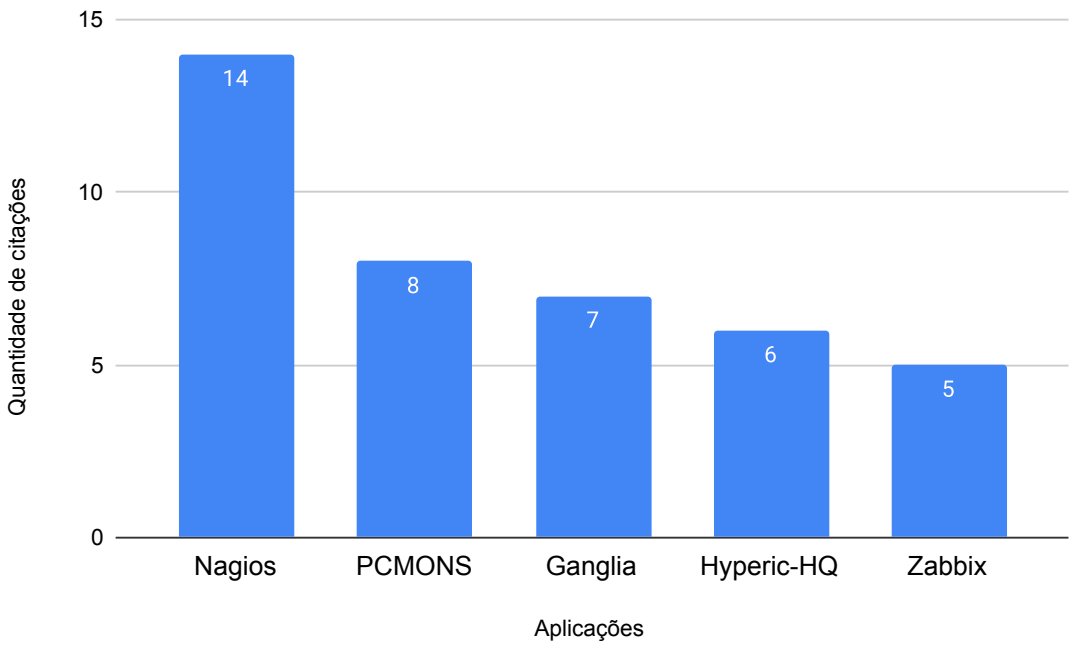
Existem ferramentas open source que são amplamente usadas no meio cientifico e algumas para finalidades corporativas. Um exemplo de ferramenta muito utilizada em meio corporativo

é o Zabbix (ZABBIX, 2020), que é uma ferramenta de código aberto para monitoramento de infraestruturas, rede, computadores e aplicativos. Criado em 2001 por Alexei Vladishev, suporta milhares de dispositivos com a descoberta automática e fornece agentes para vários tipos de Sistemas operacionais. Entretanto, também pode trabalhar sem agente.

O Hyperic-HQ é outra ferramenta para monitoramento de nuvem de código aberto que fornece dados detalhados de vários sistemas operacionais. Segundo (FATEMA et al., 2014), nos testes realizados com a ferramenta Hyperic, a mesma foi classificada como ferramenta de propósito geral em termos de monitoramento em nuvens, e, apesar de executar melhor que outras ferramentas, foi superada pelo IBM Tivoli (IBM. . . ,).

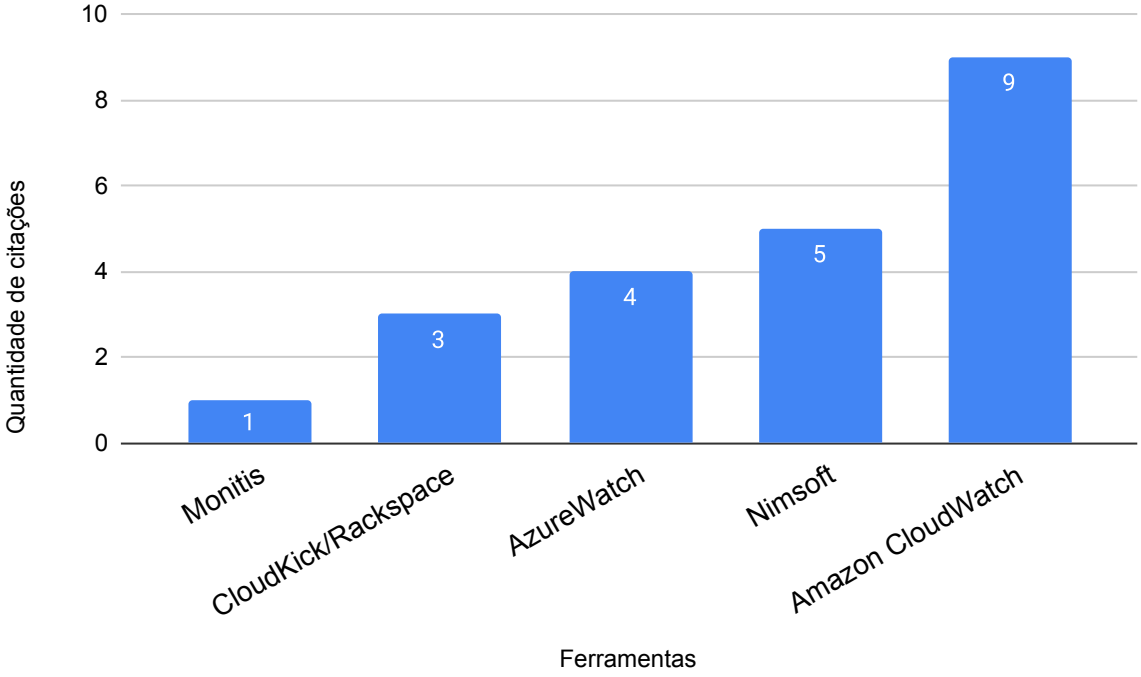
O Ganglia (GANGLIA, 2016) é um sistema de monitoramento distribuído e escalável para sistemas de computação de alto desempenho, como *clusters* e *Grids*. Baseado em uma arquitetura hierárquica, utiliza tecnologias como XML para representar os dados, XDR para transporte dos dados e RRDtool para armazenagem e visualização dos mesmos.

Figura 10 – Ferramentas Open Source



Além das ferramentas de finalidade científica acima mencionadas, ferramentas de monitoramento corporativo serão discutidas. Na Tabela 10, são apresentadas as ferramentas mais utilizadas. Foi observado que as empresas estão usando mais de uma solução de monitoramento para a infraestrutura de nuvem (SYED et al., 2017).

Figura 11 – Ferramentas Corporativas



A ferramenta Monitis (MONITIS, 2007) monitora uma variedade de serviços de TI, como sites, servidores, aplicativos, redes e instâncias virtuais de nuvem. Iniciada em 2006 com o objetivo de prover soluções de monitoramento sem agente, foi modificada depois de ser comprada pelo TeamViewer (TEAMVIEWER, 2005). Assim, ampliou para monitoramento de sites, tempo de atividade e até transações. O Monitis monitora com métricas de CPU, memória e processos, além dos serviços de rede. Contempla monitoramento em nuvens, como Microsoft Azure. Tem a API para integração, assim facilitando a personalização das necessidades de cada usuário.

A História da "Rackspace Cloud Monitoring"começa com a compra da Cloudkick em 2010. Logo após, em 2012, a Rackspace anunciou o fim da Cloudckick, substituindo pela Rackspace Cloud Monitoring API, que usa uma linguagem especifica de domínio (DSL), a qual torna a ferramenta eficaz para configurações de monitoramento avançado. Fornece suporte ao monitoramento de data centers baseados em nuvem com parâmetros mutáveis pelo consumidor para alertas, gera alarmes e notificações com granularidade opcional e fornece uma variedade de dados métricos. Os dados métricos começam em resolução cheia e, após um tempo, esses dados são convertidos para *coarse-grained*, mas para a opção do cliente pode ser *fine-grained* também. Rackspace Cloud Monitoring possui rápida capacidade de notificação, incluindo SMS(Short Message Service).

Nimsoft(NIMSOFT, 1998) foi originalmente fundada em 2004 depois da junção da NimbusSoftware e Converse Software. Em 2010, CA Technologies trouxe Nimsoft, que é uma ferramenta comercial de monitoramento com algumas características interessantes. Nimsoft pode

monitorar ampla variedade de Tecnologias da Informação, incluindo: servidores baseados em Linux, Unix, Microsoft e Cisco; ambientes na nuvem, incluindo AWS, Google Apps, Rackspace, Citrix, e Salesforce.com; e ambientes VoIP criados pela Cisco. A Nimsoft fornece informações unificadas sobre Google App, Amazon, RackSpace e Serviços baseados em Salesforce, sendo adequada para ambas infraestruturas (privadas e públicas).

AzureWatch da Microsoft é uma ferramenta flexível de monitoramento em nuvem baseado em Windows, que foi desenvolvida usando o SDK(Software Development Kit) do Windows Azure. Suporta agentes internos definidos pelo usuário para o monitoramento de recursos da nuvem.

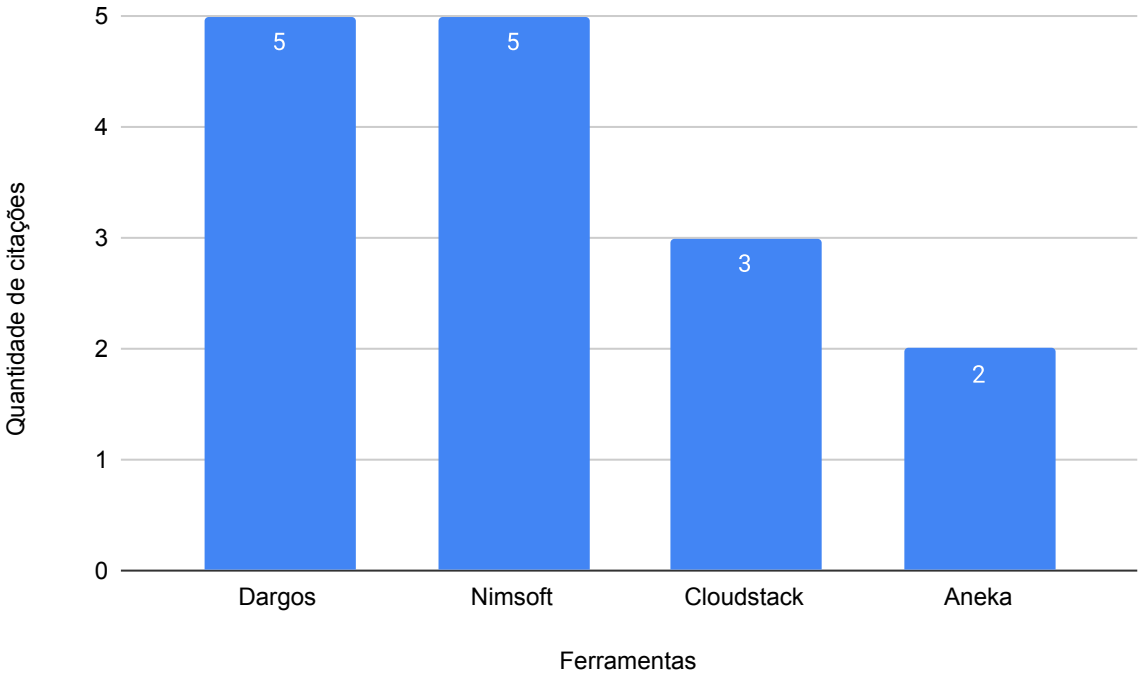
CloudWatch é uma ferramenta comercial para o monitoramento em alto nível em dados da nuvem. Amazon é o fornecedor comercial desta ferramenta, a qual pode ser usada para monitorar inúmeros serviços, como EC2. Os dados coletados pelo CloudWatch podem ser usados para análises de estatística para melhoramento da performance dos recursos de nuvem.

Questão de Pesquisa QP2

Existem ferramentas que possuem funcionalidades para monitoramento de infraestruturas híbridas (pública e privada)? Caso existam, quais são elas e quais suas vantagens e desvantagens?

As ferramentas citadas na questão de pesquisa QP1 podem ser utilizadas tanto em ambientes convencionais, para os quais foram planejadas, como em ambientes virtuais. No entanto, com o crescimento da infraestrutura virtual, surge a necessidade de ferramentas para nuvens híbridas, já que algumas ferramentas são especificas para ambientes em nuvem pública ou privada. Alguns projetos de pesquisa visualizaram esta necessidade, como o Dargos (**??**), uma arquitetura de monitoramento de nuvem distribuída. Outras ferramentas com potencial para adaptar-se em nuvens híbridas estão apresentadas na Figura 12

Figura 12 – Ferramentas híbridas



O DARGOS tem flexibilidade de monitoramento, podendo ser definida pelo usuário de acordo com sua necessidade. Os autores (POVEDANO-MOLINA et al., 2013) forneceram alguma comparação com outras arquiteturas de monitoramento de última geração. A arquitetura proposta DARGOS foram implementada como um protótipo no OpenStack. Sua implementação foram realizada usando uma rede que oferece topologia plana, *DHCP* plana e *VLAN DHCP*. No entanto, o trabalho proposto só funciona no nível L2 e suporte de rede L3. Além disso, esse cenário se torna mais difícil e cria congestionamento de rede para ambientes de vários locatários, enquanto considera acima de 50*VLANs*.

O Nimsoft é uma ferramenta de uso comercial, escalável, portátil, que utiliza o sistema de mensagem próprio, implementada em C/C++, JAVA, Perl, VB e .Net. Porém, não suporta a verificação de conformidade com SLA (Service Level Agreement - Acordo de Nível de Serviço) para localização dos dados e não possui tolerância a falhas.

O CloudStack (APACHE. . . , 2010) é uma ferramenta de monitoramento de nuvem de código aberto, sendo que sua grande ajuda é na implantação e gerenciamento de grandes recursos virtuais. Pode ser implementado via *API RESTful*, uma interface web ou uma ferramenta baseada em comandos. Semelhante ao Dargos, o CloudStack fornece informações sobre recursos. Utilizado por usuários como o AutoDesk, Dell, Huawei e Walt Disney.

A Aneka (ANEKA. . . , 2009) consiste em uma plataforma e*framework* de nuvem escalável que é implementado sobre recursos de computação heterogêneos, coordenando a execução de aplicativos, monitorando o status da nuvem e fornecendo integração com as tecnologias de

nuvem existentes. A Aneka fornece uma API extensível para o desenvolvimento de aplicativos distribuídos e suporte a diferentes tipos de nuvens: pública, privada e híbrida. A Aneka implementa uma arquitetura orientada a serviços (SOA) e os serviços são os componentes fundamentais.

Questão de Pequisa QP3

Quais são os tipos de arquitetura de monitoramento para infraestruturas virtualizadas encontrados na literatura?

Com base no artigo de (ALCARAZ CALERO; Gutiérrez Aguado, 2015), que analisa e compara arquiteturas para monitoramento de infraestruturas de computação em nuvem, especificamos as arquiteturas encontradas nos artigos da revisão sistemática, que constam na Tabela 9. Devido ao modelo de arquitetura monitorar internamente os recursos, é qualificado como Arquitetura de Monitoramento Interno (IMA), pelo fato do seu posicionamento interno. Já a Arquitetura de Monitoramento Externo (EMA) é definida quando diz sobre uma arquitetura que possui seus serviços de monitoramento fora da rede interna. Essa definição inclui serviços de monitoramento implantados dentro de VMs ou em um local externo, fora da infraestrutura de computação em nuvem (ALCARAZ CALERO; Gutiérrez Aguado, 2015).

A arquitetura Tradicional de Monitoramento IMA é o primeiro tipo. O monitoramento sem agentes ou com agente instalado nos dispositivos é permitido devido ao fato de que o provedor de nuvem tem o controle completo sobre todas as máquinas envolvidas. Existem soluções de monitoramento de código aberto de classe empresarial, como o Nagios e o Ganglia, que podem ser usados para essa finalidade. O gerenciamento não é acessível publicamente neste tipo de arquitetura, e somente o provedor de nuvem pode ter acesso a esse painel de gerenciamento. A arquitetura tradicional de monitoramento, que visa monitoramento de máquinas tradicionais somente, não foi encontrada nas pesquisas citadas pela abordagem de computação em nuvens.

A Arquitetura de Monitoramento Interno Estendido (*Extended IMA*), embora seja uma arquitetura bem utilizada, não abrange uma variedade de métricas. Assim, para ter um complemento das métricas, é necessário utilizar de ferramentas para estender as funcionalidades, como o *Hypervision* (POPEK; GOLDBERG, 1974), plug-in que reúne outras métricas por meio do monitoramento remoto sem agente das VMs (*virtual Machine*)(ALCARAZ CALERO; Gutiérrez Aguado, 2015). Alguns trabalhos utilizaram este tipo de arquitetura, como o DARGOS (**??**), (ZOU et al., 2013), (XU; CHEN; Alcaraz Calero, 2017) e o (CARVALHO, 2015).

Por alguns motivos, as abordagens tradicionais de monitoramento não se encaixam bem no monitoramento de infraestruturas de computação em nuvem. A principal razão é a incrível diferença no ciclo de vida das VMs em relação às máquinas físicas. As infraestruturas físicas raramente alteram sua topologia, principalmente pelos risco de falhas ou substituições de hardware. Além disso, a infraestrutura física geralmente mantém constante o número de serviços prestados. Esses recursos foram a base durante o desenvolvimento de todos os serviços em torno do ecossistema de monitoramento tradicional: agentes de monitoramento, protocolos

de descoberta automática de hardware e serviços, etc.

No entanto, infraestruturas virtuais expõem um ambiente completamente diferente. Primeiramente, os serviços precisam ser monitorados de maneira transparente. Segundo, as VMs podem ser removidas, causando mudanças frequentes na topologia e os protocolos de descoberta atuais não reagem adequadamente. Terceiro, endereços IP são reutilizados constantemente, causando incoerência de informações devido a conflitos de IP. Para solucionar esses problemas, é necessário criar um módulo de monitoramento especifico para componentes "voláteis"da infraestrutura. Essas alterações são notificadas ao serviços assim que detectadas, permitindo que um serviço de monitoramento seja auto-adaptável. A arquitetura que inclui esta nova função é referida como Arquitetura de Monitoramento Interno Estendido e Adaptativo (Extended and Adaptive IMA).

Tanto o Extended IMA quanto o Extended and Adaptive IMA são adequados para fornecer os serviços de monitoramento ao consumidor da nuvem. O único requisito extra sobre essas arquiteturas é que o serviço de monitoramento deve fornecer suporte a vários provedores para permitir que todos os consumidores da nuvem acessem sua própria visão geral da infraestrutura. De fato, o IMA Estendido e Adaptável pode ser estendido opcionalmente com o uso de um agente de software instalado na máquina virtual do consumidor da nuvem, a fim de fornecer informações de monitoramento mais detalhadas ao consumidor.

Como esta arquitetura consiste em adaptação e extensão, se torna a mais utilizada entre os trabalhos selecionados na revisão. São 17 que seguem o padrão de arquitetura descrito acima: (PEREZ-ESPINOZA et al., 2016), (KATSAROS et al., 2012), (SUN et al., 2010), (SYED et al., 2018), (MORADI et al., 2017), (CIUFFOLETTI, 2016), (SMIT; SIMMONS; LITOIU, 2013), (MONTES et al., 2013), (AKTAS, 2018a), (GUTIERREZ-AGUADO; Alcaraz Calero; Diaz Villanueva, 2016), (LU et al., 2016), (De Chaves; URIARTE; WESTPHALL, 2011), (EMEAKAROHA et al., 2012), (KUBLER; MINOR, 2015), (Currie A.R., McConnell A., Parr G.P.; K., 2014), (SCHUBERT, 2014), (SÁ, 2013).

A arquitetura de Monitoramento Externo Concentrado (Concentrated EMA) permite que o consumidor tenha um módulo de monitoramento mais preciso, acessado de forma pública pois possui IP público, e os clientes possam acessar a interface de monitoramento e gerenciamento. Fornece um isolamento completo dos serviços de monitoramento entre diferentes consumidores de nuvem. Somente dois utilizaram este tipo de arquitetura: o MonPaas (ALCARAZ CALERO; AGUADO, 2015) e o Scalable Agentless (BRATTSTROM; MORREALE, 2017).

Portanto, Extended and Adaptive IMA se torna uma boa opção dever-se-á ao fato que tanto cliente quanto provedor usam o mesmo serviço de monitoramento. Entretanto, somente em cenários em que existe um isolamento entre provedor e consumidor, pela questão de segurança de informações de monitoramento. Assim sendo, então, pode ser valioso usar esse modelo em arquiteturas híbridas, porém a sobrecarga pode ser maior com esse isolamento entre cliente e CSP.

Tabela 9 – Arquiteturas de monitoramento dos trabalhos selecionados

Artigo com Citação	**Arquitetura**
(PEREZ-ESPINOZA et al., 2016)	Extended and Adaptive IMA
(KATSAROS et al., 2012)	Extended and Adaptive IMA
(SUN et al., 2010)	Extended and Adaptive IMA
(SYED et al., 2018)	Extended and Adaptive IMA
(MORADI et al., 2017)	Extended and Adaptive IMA
(POVEDANO-MOLINA et al., 2013)	Extended IMA
(ZOU et al., 2013)	Extended IMA
(CIUFFOLETTI, 2016)	Extended and Adaptive IMA
(XU; CHEN; Alcaraz Calero, 2017)	Extended IMA
(SMIT; SIMMONS; LITOIU, 2013)	Extended and Adaptive IMA
(MONTES et al., 2013)	Extended and Adaptive IMA
(AKTAS, 2018a)	Extended and Adaptive IMA
(GUTIERREZ-AGUADO; Alcaraz Calero; Diaz Villanueva, 2016)	Extended and Adaptive IMA
(LU et al., 2016)	Extended and Adaptive IMA
(ALCARAZ CALERO; AGUADO, 2015)	Concentrated EMA
(BRATTSTROM; MORREALE, 2017)	Concentrated EMA
(De Chaves; URIARTE; WESTPHALL, 2011)	Extended and Adaptive IMA
(EMEAKAROHA et al., 2012)	Extended and Adaptive IMA
(KUBLER; MINOR, 2015)	Extended and Adaptive IMA
(Currie A.R., McConnell A., Parr G.P.; K., 2014)	Extended and Adaptive IMA
(CARVALHO, 2015)	Extended IMA
(SCHUBERT, 2014)	Extended and Adaptive IMA
(SÁ, 2013)	Extended and Adaptive IMA

Questão de Pequisa QP4

Quais são as principais métricas ou informações monitoradas em infraestruturas computacionais virtualizadas?

As métricas monitoradas mais frequentes são os recursos da máquina (CPU, Memória, disco e rede), seja ela física ou virtual, conforme a Tabela 3.4.1. Em alguns casos, são utilizados sensores capazes de obter dados do sistema, do banco de dados, aplicativos e serviços. As ferramentas de monitoramento são usadas há muito tempo para rastrear a utilização de recursos e

o desempenho de sistemas e redes. Dos trabalhos citados na Tabela 3.4.1, 17 adotam métricas voltadas para monitoramento de recursos.

Como o provedor é responsável pela manutenção da infraestrutura física em modelo de computação em nuvem, um aspecto atraente das ferramentas é o monitoramento de performance, já que em algum momento o desempenho de algum servidor pode ficar degradado. Para que o SLA não seja violado, é importante monitorar o desempenho da infraestrutura física.

Outro ponto muito importante que não é implementado em uma grande parcela dos trabalhos vistos é o monitoramento de faturamento, ou seja registrar os recursos gastos ou que passaram do orçamento previsto. Em nuvens públicas, como a Amazon Web Services (AWS), Microsoft Azure e a Google Cloud Platform, existe monitoramento dos recursos usados e formas de criar alertas para itens que passarem do gasto pré-definido.

Tabela 10 – Métricas e nuvens dos trabalhos selecionados

Métricas monitoradas	**Nuvem privada**	**Nuvem pública**	**Referência**
Performace	X		(PEREZ-ESPINOZA et al., 2016)
Performace e recursos	X	X	(KATSAROS et al., 2012)
Recursos		X	(SUN et al., 2010)
Recursos	X		(SYED et al., 2018)
Recursos	X	X	(MORADI et al., 2017)
Performace	X	X	(POVEDANO-MOLINA et al., 2013)
Recursos e processos		X	(ZOU et al., 2013)
Performace	X	X	(CIUFFOLETTI, 2016)
Recursos	X	X	(XU; CHEN; Alcaraz Calero, 2017)
Recursos	X	X	(SMIT; SIMMONS; LITOIU, 2013)
Performace e faturamento	X		(MONTES et al., 2013)
Recursos	X	X	(AKTAS, 2018b)
Recursos	X	X	(GUTIERREZ-AGUADO; Alcaraz Calero; Diaz Villanueva, 2016)
Recursos e processos	X	X	(LU et al., 2016)
Performace e recursos	X	X	(ALCARAZ CALERO; AGUADO, 2015)
Recursos	X	X	(BRATTSTROM; MORREALE, 2017)
Recursos e perfomace	X		(De Chaves; URIARTE; WESTPHALL, 2011)
Recursos	X	X	(EMEAKAROHA et al., 2012)
Performace	X		(KUBLER; MINOR, 2015)
Recursos		X	(Currie A.R., McConnell A., Parr G.P.; K., 2014)
Recursos	X	X	(CARVALHO, 2015)
Recursos e faturamento	X	X	(SCHUBERT, 2014)
Recursos	X	X	(SÁ, 2013)

Questão de pesquisa complementar

QP5 Quais artigos/trabalhos abordam monitoramento de aplicações do tipo API RESTful?

Com base nos estudos realizados em torno das outras questões e a observação perante ao crescente uso de serviços na Internet na forma de aplicações web e um amplo movimento em direção ao RESTful (NEUMANN; Laranjeiro; Bernardino, 2018), visualizamos a necessidade por monitoramento dessas aplicações. Com o comum uso de recursos em nuvens, um método popular é a projeção de aplicativos em arquitetura de microsserviços, pois possuem uma abordagem escalável baseada em nuvem (SAMPAIO et al., 2019). Esses aplicativos em microsserviços comumente utilizam da comunicação síncrona por meio de APIs REST, a arquitetura de microsserviço e também podem usar comunicação assíncrona por meio de protocolos Pub-Sub.

Foram realizados diversos mapeamentos e outras pesquisas à parte da revisão sistemática sobre o monitoramento de aplicações do tipo API RESTful, porém não encontramos trabalhos sobre o tema em específico. Entretanto, encontramos trabalhos com estudos relacionados a comunicação por meio de APIs REST, porém nenhum que trate o monitoramento delas em especifico.

Alguns sistemas de monitoramento utilizam o paradigma REST e outros utilizam URIs para endereçar entidades, que é o caso da API OCCI utilizada na pesquisa (CIUFFOLETTI, 2016). Em outro artigo (AL-SHAMMARI; HUSEIN, 2020), foi utilizada a arquitetura *Representational State Transfer* (REST) para um sistema de monitoramento, apresentando uma avaliação de desempenho dos dados transmitidos. Na seção 4.4 de trabalhos relacionados, abordamos sobre.

3.5 Limitações desta revisão

Em estudos de revisão, sistemáticos ou não, é comum que algumas limitações sejam apresentadas (PETERSEN, 2008). Esta revisão apresenta algumas das limitações mais comuns apresentadas por estes tipos de estudos, como a formulação limitada das questões de pesquisa e a orientação de tendências a publicações e seleção de imprecisões durante a extração de dados.

Com o intuito de minimizar essas limitações, usamos critérios bem direcionados para nosso protocolo de pesquisa. A seguir, abordamos essas limitações e a estratégia para mitigar cada uma delas. Primeiro, a formulação das perguntas de pesquisa que não podem fornecer uma cobertura completa do campo para monitoramento de TI de forma geral. Para diminuir essa ameaça, analisamos previamente a literatura, com autores e especialistas para verificar as dúvidas. Dessa forma, selecionamos, dentro do possível, o melhor conjunto de perguntas. Segundo, a condução da pesquisa pelas bases de dados digitais utilizadas neste estudo não possuem regras de pesquisa compatíveis. Há uma possibilidade que alguns estudos tenham sido perdidos por

empregarem palavras-chaves sinônimas às utilizadas na busca. Como forma de mitigar esses problemas, em cada base de busca digital ajustamos as nossas strings de busca, porém mantendo os mesmos termos. Também efetuamos combinações de busca automática e manual.

3.6 Considerações finais sobre a Revisão

Essa revisão teve como objetivo identificar os trabalhos do estado da arte que abordassem a utilização de monitoramento de ambientes em nuvem. A revisão tem uma grande contribuição para identificar as lacunas, trazer o resumo de evidências dos trabalhos, reforçando toda a análise perante ao tema proposto na dissertação. Para isso, foi utilizado o método de revisão sistemática, que auxiliou no processo de busca, seleção e filtragem de artigos. No final, obteve-se a quantia de 23 estudos relevantes. A partir da análise dos artigos, foi possível analisar a evolução das pesquisas na área, apresentar as ferramentas e arquiteturas utilizadas, e seus principais objetivos e para onde estão sendo direcionados.

4

Trabalhos Relacionados

Neste capítulo são abordados os trabalhos relacionados à proposta. Alguns trabalhos apresentados nesta seção também são resultantes da revisão sistemática da seção 3. Além dos trabalhos que foram apresentados como resultados da revisão, existem outros que são relacionados com a proposta de alguma forma, seja pelo uso do monitoramento em infraestruturas virtuais em nuvens públicas e privadas ou pelo desenvolvimento de uma arquitetura para monitoramento de aplicações API REST, monitoramento de infraestrutura e IoT (Internet das coisas), além do monitoramento comum de dispositivos de rede. Nesta seção, apresentamos os principais trabalhos que possuem alguma similaridade com o atual, seja pelo uso do monitoramento em aplicações ou pelo desenvolvimento de uma arquitetura para monitoramento das infraestruturas, e algumas revisões sobre o tema monitoramento para disseminação de informações neste âmbito.

4.1 Revisões e mapeamentos da literatura

A maioria dos trabalhos encontrados em nossa revisão da literatura sobre monitoramento abordam sobre infraestruturas em nuvens. Alguns trabalhos mostram uma revisão do tema, principalmente direcionados para nuvens, infraestruturas virtuais e IoT (Internet of Things), como no estudo realizado por Fatema et al. (FATEMA et al., 2014), mostrando uma visão para uma revisão de ferramentas de monitoramento na nuvem, examinando, especificamente, ferramentas da nuvem, com detalhes técnicos. Nesta seção, é detalhada a metodologia da arquitetura dos sistemas de monitoramento, esmiuçando os componentes monitorados, a comunicação com o servidor de monitoramento, banco de dados e as informações e alertas gerados pelos recursos monitorados.

Outros trabalhos mostram uma revisão do tema monitoramento, como no estudo (ACETO et al., 2013), o qual aborda o crescimento do ambiente na nuvem e os fatores com base na análise da literatura. Em outro trabalho do mesmo autor no ano anterior (ACETO et al., 2012), são

tratadas as atividades no ambiente de nuvem que tem necessidade por monitoramento. Neste ano, Aceto e outros (2012) citaram que as plataformas atuais não cumprem todos os requisitos expressos em documento.

Gill possui um outro exemplo de trabalho (GILL; HEVARY, 2016) que adota uma abordagem de revisão sistemática para identificar os desafios de monitoramento. Em seus resultados, foram identificados cinco principais desafios dos dados de monitoramento de nuvem: tecnologia de monitoramento, tecnologia de virtualização, energia, disponibilidade e desempenho. Esses trabalhos ajudam na visão da área de monitoramento em nuvem. Um dos principais desafios apresentados nessa pesquisa é a falta de padronização dos dados, sendo que esta falta de padrões impede a capacidade de integrar ferramentas de monitoramento. Uma das soluções é centralizar as informações em uma ferramenta, que possa ter *plug-ins* de outras aplicações, assim como a pesquisa apresentada por nós neste documento, em que nos aprofundamos no âmbito do monitoramento e trazemos outras possibilidades em monitoramento de aplicações API RESTful e de nuvens.

As revisões e mapeamentos sistemáticos citados contribuíram para levantar informações na seção de revisão sistemática 3, como a extração das ferramentas usadas em nuvens, as arquiteturas encontradas na literatura e as principais métricas monitoradas. Após selecionar as questões e avaliar todos os estudos, verificamos que há necessidade por outros tipos de monitoramentos, que abrangem o monitoramento de nuvens e de ambientes *on-premises*. Após os resultados das questões, observamos essa lacuna para monitoramento de aplicações API RESTful, e uma pesquisa complementar foi realizada para levantar os dados sobre.

4.2 Monitoramento de nuvens privadas

Em toda pesquisa realizada, poucos foram os trabalhos que encontrados específicos para nuvens privadas. A maioria dos trabalhos são para infraestruturas híbridas.

Em um estudo (De Chaves; URIARTE; WESTPHALL, 2011) foi elaborado um sistema de monitoramento para nuvens privadas, PCMONS (Private Cloud MONitoring System). A arquitetura do projeto é composta por camadas, sendo elas: infraestrutura, integração e visão. A camada de infraestrutura contém a diferença das plataformas Eucalyptus (EUCALYPTUS. . . , 2008) e OpenNebula (OPENNEBULA, 2008) e as tecnologias de virtualização Xen (XEN,) e (KERNEL-BASED. . . , 2007). A camada de integração possui uma interface que integra o acesso às informações, um acesso comum para abstrair a diferença na camada de infraestrutura. Já a camada de visão utiliza a interface anterior para oferecer os dados comuns para visualização, de acordo com o público.

Neste artigo (PEREZ-ESPINOZA et al., 2016) é proposta uma arquitetura distribuída e modular para nuvens privadas. A arquitetura de monitoramento possui os seguintes módulos: Coletor, Meta sensor, Distribuidor e Visualizador. Os módulos foram projetados para serem

independentes e funcionarem de maneira integrada no modelo cliente/servidor. Um dos pontos tratados é a escalabilidade e tolerância a falhas. A coleta é distribuída, extraindo as informações de status dos diferentes recursos monitorados. Após o mapeamento dos recursos monitorados pelos coletores, o distribuidor concentra as informações de status. A solução do projeto é reduzir a carga de trabalho dos recursos monitorados. A distribuição impede a centralização de dados, balanceando a carga e permitindo a tolerância a falhas.

Como os dois trabalhos acima são projetados para nuvens privadas, mostra-se uma desvantagem para expansão do sistema de monitoramento, caso seja necessário abranger para nuvens públicas e transformar a infraestrutura virtual em nuvens híbridas. Outro ponto analisado é a abordagem para ferramentas especificas, podendo causar aprisionamento.

4.3 Monitoramento de Nuvens Hibridas

No levantamento realizado na revisão sistemática 3, dos 23 trabalhos selecionados, a maioria das pesquisas que foram para nuvens híbridas, descritas na tabela 3.4.1, serviam tanto para públicas como para privadas.

No trabalho de Calero (ALCARAZ CALERO; Gutiérrez Aguado, 2015) foi realizada uma comparação entre arquiteturas, abordando alguns modelos, como explanamos na questão de pesquisa 3.4.1 e classificamos as pesquisas conforme este trabalho. Na tabela 9 é mostrada, com detalhes, toda a composição e funcionalidades, que possui tanto uma analise com um número significativo de diferentes cenários de monitoramento como também em ambientes de nuvens híbridas. Este trabalho teve uma grande relevância para segmentação e entendimento sobre os modelos de arquiteturas e levantamento dos outros trabalhos selecionados na revisão.

Em outra pesquisa realizada por Alcaraz Calero (ALCARAZ CALERO; AGUADO, 2015), foram identificados os principais desafios associados à computação em nuvem. Além disso, foi apresentada uma abordagem sobre uma arquitetura de monitoramento que fornece uma visão geral do status da infraestrutura de computação por meio de uma plataforma integrada com a ferramenta Nagios à plataforma de computação em nuvem (OPENSTACK. . . , 2010). Para monitorar, o MonPaas analisa as comunicações entre os componentes do OpenStack diretamente no barramento de mensagens, realizando as chamadas para API REST (*Representational State Transfer*) do NConf (NCONF,), que é uma interface baseada na Web para executar a configuração do Nagios. Com a visão de escalabilidade, é utilizado, também, o DNX[1], uma outra extensão do Nagios que permite a distribuição da plataforma de monitoramento, resolvendo, assim, um problema anterior do IaaSMon (GUTIERREZ-AGUADO; Alcaraz Calero; Diaz Villanueva, 2016) de falta de informação e controle dos consumidores de infraestruturas em nuvem. O trabalho tem uma arquitetura distribuída e recursos adaptativos ou, também, manuais. O sistema propõe uma autonomia para personalizar e configurar as informações coletadas.

[1] http://dnx.sourceforge.net/

Este trabalho tem um passo significativo no campo do monitoramento, sendo uma continuação do IaaSMon (GUTIERREZ-AGUADO; Alcaraz Calero; Diaz Villanueva, 2016). É, também, uma solução gratuita e de código aberto, que, segundo o autor, foi um complemento do projeto.

Algumas soluções abordadas trabalham com a proposta de plug-ins e trabalham integradas a grandes ferramentas para atender uma infraestrutura ou objetivo específico de deficiência em uma arquitetura. O Rocmon é a proposta do trabalho de (CIUFFOLETTI, 2016), que utiliza o Nagios como ponto de partida. O sistema de monitoramento resume a experiência do Nagios com uma perspectiva de nuvem. O Rocmon é orientado a plug-ins. O sistema de monitoramento Rocmon adota uma extensão da OCCI (Open Cloud Computing Interface), uma API definida na estrutura do OGF (Open Grid Forum). A extensão de monitoramento OCCI permite que o usuário defina e solicite, como serviço, a implementação de uma infraestrutura de monitoramento, cobrindo os recursos obtidos do provedor de nuvem.

No artigo de (BRATTSTROM; MORREALE, 2017), é abordada uma proposta de monitoramento de infraestrutura sem agente, no qual é utilizado o protocolo SNMP para obter as métricas a nível de rede. Os dados são colocados no banco Prometheus e apresentados no Grafana. Como o nosso sistema de monitoramento proposto, esse também é uma alternativa de código aberto para monitoramento. No caso deste artigo (BRATTSTROM; MORREALE, 2017), a visão é focada para recursos de rede, que são obtidos pelo protocolo SNMP, contando com um painel que fornece dados de monitoramento, que não adiciona sobrecarga ou impacto no desempenho da rede. O artigo traz uma visão de escalabilidade de um monitoramento sem agente, podendo serem adicionados vários pontos de monitoramento com um Raspberry Pi com o sistema incluso. Na nossa proposta, existe uma semelhança perante a escalabilidade: o sistema necessita de alguns contêineres caso seja necessário escalar o monitoramento da nuvem e das respectivas aplicações API RESTful.

Na pesquisa de (SYED et al., 2018), é citada uma estrutura de monitoramento de desempenho que visa resolver os desafios de monitoramento não-intrusivo, podendo funcionar em todos os ambientes do provedor com menos sobrecarga no computador físico, com uma estrutura horizontal e vertical escaláveis, em que escalabilidade vertical é obtida através da coleta eficiente de métricas de monitoramento do sistema operacional, e a escalabilidade horizontal é obtida por meio do mecanismo de envio, em que os nós físicos transferem as informações coletadas para o nó central do controlador da nuvem. Para obter os dados de monitoramento, o CloudProcMon (SYED et al., 2018) utiliza o (NRPE) *Nagios Remote Plug-in Executor*, que coleta os dados vinculados ao identificador de VM.

A arquitetura de software de monitoramento de nuvem híbrida (AKTAS, 2018a) foi criada para funcionar como uma camada complementar na parte superior das plataformas de computação em nuvem existentes. A proposta é projetar a arquitetura com base em um padrão de design de software simples, utilizando um conceito complexo de processamento de eventos, no

qual dados de vários métricas são processados, afim de detectar padrões previamente definidos.

O JTangCMS (sistema de monitoramento de nuvem JTang) (LU et al., 2016) é um sistema de monitoramento para plataformas de nuvem, que abrange a coleta, entrega e processamento de dados de monitoramento. Para coleta de dados, possui um agente que é considerado flexível e escalonável, implementado com plug-ins de monitoramento que coletam as informações de tempo de execução. Para entrega de dados, uma estrutura de disseminação de dados eficiente e robusta foi implementada para transmitir as informações de tempo de execução com alta taxa de transferência e baixa latência, com base no (DDS) *Data Distribution Service*. Para o processamento de dados, uma plataforma de ação na nuvem é desenvolvida para suportar a tomada de decisões de gerenciamento de nuvem, com base no (CEP)*complex event processing*.

O Dargos (POVEDANO-MOLINA et al., 2013) propõe um sistema de monitoramento centralizado, utilizando um sistema chamado DDS que disponibiliza os dados de monitoramento, como no projeto (LU et al., 2016). O DDS é baseado em *publish/subscribe*, que possuem agentes (*publishers*) responsáveis por coletar dados de monitoramento computacionais, fazendo com que estes dados sejam enviados para entidades (*subscribers*), realizando o registro do monitoramento. As entidades citadas podem ser quaisquer módulos ou sistemas que dependam de dados do monitoramento para seu funcionamento. Como exemplo, módulos para aprovisionamento de recursos, agendadores ou interfaces de monitoramento e de administração. Assemelha-se com as outras plataformas de monitoramento citadas anteriormente.

GMonE (MONTES et al., 2013) é uma solução de monitoramento modular. Assim como o Dargos (POVEDANO-MOLINA et al., 2013), utiliza comunicação *publish/subscribe*. A ferramenta envia as medições de monitoramento dos módulos GMonEMon para serem armazenadas pelo módulo de banco GMonEDB. Desenvolvido em Java, estes plug-ins podem ser estendidos para os módulos GMonEMon, o qual é responsável pela obtenção de coleta. O módulo GMonEAccess permite a visão dos dados que estão armazenados no GMonEDB. Para adicionar novos módulos, é necessário desenvolvimento da ferramenta, no caso desenvolvida em Java. Como demandará tempo para aprendizagem, torna-se uma desvantagem em relação a outras aplicações que possuem módulos e/ou plug-ins para expansão.

As arquiteturas citadas acima são sistemas robustos, em que alguns deles precisam de ferramentas terceiras para complementar o monitoramento. Outros possuem suas próprias soluções, que podem elevar o tempo para adaptação no ambiente, caso seja necessário aplicar outros plug-ins e/ou soluções para o escopo necessário. Todos os trabalhos podem ser utilizados em ambas as nuvens (publica e privada). Esse é um ponto interessante, pois é o mesmo objetivo da nossa proposta. Com a utilização de alguns softwares terceiros, que podem ser substituídos por outros existentes, propusemos uma arquitetura integrável e baseada em plug-ins e desenvolvemos um plug-in para abranger o tipo de monitoramento, que pode incluir outros dispositivos que utilizem comunicação API Rest.

4.4 Outros tipos de monitoramentos

Nos estudos feitos, não foram encontrados direcionamentos para monitoramento de aplicações do tipo API RESTful. Alguns trabalhos citam ferramentas que podem ser usadas nestes aspectos, porém nenhum trabalho direcionou estudos ou propôs novas abordagens sobre o tema.

No artigo apresentado por (AL-SHAMMARI; HUSEIN, 2020), há uma abordagem sobre o monitoramento dos dados na IoT que requer um processamento na nuvem. Como as tecnologias de serviços da web são implementadas em sua maioria por APIs na nuvem, este artigo propõe uma implementação de um aplicativo da web em nuvem para um monitoramento inteligente usando a arquitetura REST. O artigo contém uma avaliação de desempenho desses dados transmitidos, que inclui o tempo de resposta do processo de transmissão. Esta pesquisa também analisa a sobrecarga de QoS (Quality of Service). Como esse trabalho analisa a comunicação de aplicações com a arquitetura REST, o desempenho da comunicação e o tempo de resposta desses tipos de comunicação com a nuvem se tornam um complemento para este trabalho, que monitora o status dessas APIs REST.

Poucos trabalhos abordaram somente nuvens públicas. A pesquisa de (STEPHEN; BENEDICT; KUMAR, 2018) tem a visão do desempenho e análise dos recursos disponíveis, sendo feita em instancias de nuvem da Amazon, a qual utiliza o próprio gerenciador CloudWatch e o IDERA. A pesquisa analisa e compara as ferramentas de monitoramento e mostra a importância da disponibilidade métrica dos recursos. Esta é uma das poucas pesquisas que analisam uma arquitetura de monitoramento para nuvem pública.

Em um ambiente com várias aplicações, em que pelo grande uso de microsserviços e sua natureza dinâmica, torna o monitoramento desses serviços uma tarefa desafiadora (MORADI et al., 2017). O ConMon é um sistema automatizado para monitorar o desempenho da rede de aplicativos baseados em contêineres. Identificando automaticamente os contêineres de aplicativos recém instanciados, o ConMon (MORADI et al., 2017) observa passivamente seu tráfego. Com base nessas observações, ele configura e executa funções de monitoramento dentro de contêineres de monitoramento adjacentes. O sistema adapta os contêineres de monitoramento, dependendo das mudanças ocorridas pelo aplicativo ou pela plataforma de execução.

4.5 Considerações

Considerando os avanços realizados por outros trabalhos, esse trabalho visa contribuir com uma arquitetura de monitoramento escalável e integrável. Além disso, superar algumas limitações de trabalhos anteriores, como a utilização de aplicações que possuem uma curva de aprendizagem acentuada, bem como uma falta de adaptabilidade e abrangência de monitoramento de outros recursos como o citado neste trabalho. Na Tabela 11, tem-se todos os trabalhos

relacionados. Para cada estudo, é apresentada a referência, o tipo de métrica monitorada, a estratégia de coleta dos dados, a arquitetura e a principal contribuição de cada trabalho. Desse modo, a Tabela fornece uma visão geral das principais características dos trabalhos relacionados. Ademais, no último item também é apresentado o presente trabalho. O presente trabalho aqui proposto visa contribuir com uma nova arquitetura de monitoramento de infraestrutura virtual expansível, levando em consideração os desafios e particularidades do processo das infraestruturas virtuais.

Tabela 11 – Trabalhos relacionados e suas principais características

Referência	**Arquitetura**	**Nuvem**	**Principais contribuições**
(De Chaves; URIARTE; WESTPHALL, 2011)	Extended and Adaptive IMA	Privada	Uma arquitetura flexível e extensível, o PCMONS pode ser adaptado para uso por provedores.
(PEREZ-ESPINOZA et al., 2016)	Extended and Adaptive IMA	Privada	Uma arquitetura distribuída para monitoramento de nuvens privadas, com balanceamento de carga e tolerância a falhas.
(ALCARAZ CALERO; AGUADO, 2015)	Extended and Adaptive IMA + Concentrated EMA	Híbridas	Um arquitetura de monitoramento distribuída e altamente escalável para nuvem. Com o MonPaaS, o consumidor da nuvem personaliza suas próprias métricas, serviços e recursos.
(GUTIERREZ-AGUADO; Alcaraz Calero; Diaz Villanueva, 2016)	Extended and Adaptive IMA	Híbridas	Uma solução completa de monitoramento distribuído e altamente escalável, adequada para monitorar infraestruturas de nuvem pública.
(CIUFFOLETTI, 2016)	Extended and Adaptive IMA	Híbridas	Um estudo para compreender os limites e prosseguir com o projeto de uma nova arquitetura de monitoramento utilizando o Nagios, usando integração de monitoramento com os planos de controle e dados da pilha de nuvem.
(BRATTSTROM; MORREALE, 2017)	Concentrated EMA	Híbridas	Uma abordagem de monitoramento sem agente, sem intrusão, utilizando um dispositivo Pi.

Table 11 continued from previous page

Referência	Arquitetura	Nuvem	Principais contribuições
(SYED et al., 2018)	Extended and Adaptive IMA	Híbridas	Uma abordagem de monitoramento de nuvem não intrusivo baseado em sistema operacional host chamado CloudProcMon.
(AKTAS, 2018a)	Extended and Adaptive IMA	Híbridas	Uma arquitetura de monitoramento de nuvem híbrida para detecção de defeitos e lacunas de segurança.
(LU et al., 2016)	Extended and Adaptive IMA	Híbridas	Um sistema de monitoramento de nuvem chamado JTangCMS .
(POVEDANO-MOLINA et al., 2013)	Extended IMA	Híbridas	Uma arquitetura descentralizada e flexivel DARGOS com uso de DDS e proxies que permitem ser estendidos em cenários de escala industrial.
(MONTES et al., 2013)	Extended and Adaptive IMA	Híbridas	Uma arquitetura genérica de monitoramento de nuvem, o GMonE apresenta um comportamento bom com as seguintes métricas: desempenho, escalabilidade, resolução de monitoramento e overhead.
(AL-SHAMMARI; HUSEIN, 2020)	-	-	Uma aplicação web com uso do protocolo REST para gerenciamento de um sistema de monitoramento e análise de sobrecarga do tempo de resposta.
(STEPHEN; BENEDICT; KUMAR, 2018)	-	Pública	Analise da métrica de disponibilidade, o SLA, para nuvens públicas e seu uso na visão do consumidor.
(MORADI et al., 2017)	Extended and Adaptive IMA	-	Um sistema baseado em contêiner distribuído para monitoramento automatizado
Esse estudo	**Extended and Adaptive IMA**	**Híbridas**	**Uma arquitetura com rápida inserção em uma infraestrutura existente com maior adaptação, escalável e integrável com outras aplicações de monitoramento. Monitoramento adequado para aplicações com padrões REST.**

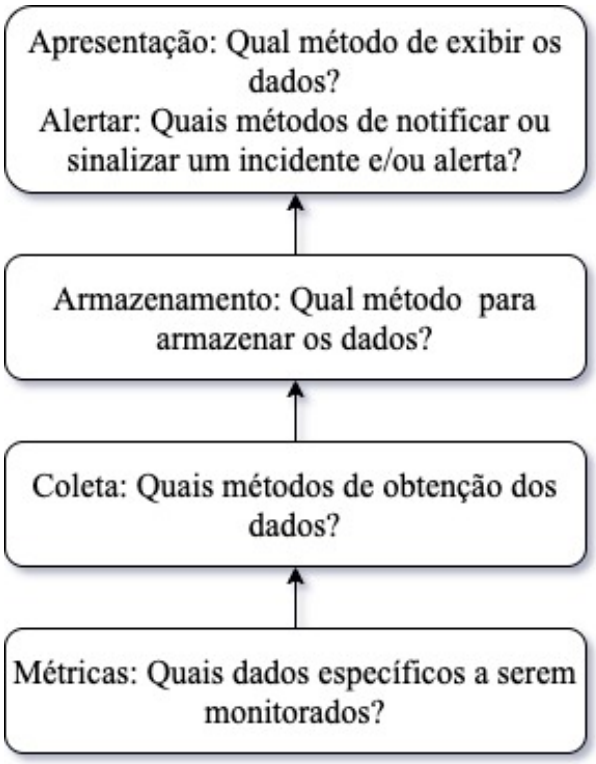
5

Arquitetura de monitoramento

5.1 Sistemas de monitoramento de infraestruturas

Os sistemas atuais de monitoramento possuem uma estrutura essencial dividida em cinco partes: métricas, coleta, armazenamento, apresentação e alerta. Para corresponder a um bom nível de monitoramento, estes parâmetros precisam ser contemplados. Para contemplar os principais requisitos de um sistema de monitoramento, podemos responder às cinco perguntas básicas presentes na Figura 13:

Figura 13 – Questões que guiam os sistemas de monitoramento.



Métricas: Quais dados específicos a serem monitorados?

Para analisar as métricas de ativos e aplicações remotas, pode-se usar dois métodos de abordagem: sem agente e com agente. Na forma de analisar métricas com agente, é necessário ter uma aplicação no dispositivo remoto para ser a forma de coletar internamente as métricas necessárias. Quando é utilizada a forma sem agente no item a ser monitorado, pode-se usar uma API ou protocolo remoto para analisar as métricas de um dispositivo ou aplicação diferente.

Dentre as métricas de monitoramento mais comuns estão os indicadores de disponibilidade (*(e.g., uptime, downtime)*), consumo de recursos (e.g., CPU, memória, uso de disco, atividade da rede), latência e desempenho (e.g., tempo de resposta, vazão, latência) (ACETO et al., 2012). Muitos sistemas de monitoramento utilizam protocolos conhecidos que geralmente são padrão na maioria das máquinas e em outros dispositivos da rede, como também protocolos no nível de aplicação como o HTTP (Hypertext Transfer Protocol). Estes protocolos são utilizados para obter as métricas de disponibilidade e até desempenho dos recursos, sendo o SNMP um dos principais protocolos para essa tarefa (BRATTSTROM; MORREALE, 2017).

Coleta: Quais métodos de obtenção dos dados?

É a forma de recuperar as métricas dos ativos monitorados, no caso do monitoramento de rede. Argumenta-se que os sistemas de monitoramento possuem duas categorias, que podem ser na forma de enviar ou extrair. O ponto que os diferencia é qual elemento inicia a comunicação para transferência dos dados monitorados: aquele que gerencia o ambiente (método*polling*), ou o próprio elemento monitorado (método *trapping*) (MAURO; SCHMIDT; LOUKIDES, 2001).

No método *polling*, o elemento gerenciador envia uma solicitação ao elemento monitorado para obter métricas em um intervalo ou evento específico. Neste caso, é útil para extrair dados específicos quando o evento é crítico. Esta técnica, geralmente, é usada via API ou protocolo remoto, podendo não haver agente do lado monitorado. No método *trapping*, o elemento monitorado envia suas próprias métricas para o elemento gerenciador em um intervalo ou evento específico. Uma das vantagens é enviar métricas e não sobrecarregar, além de reduzir o congestionamento a nível de rede. Geralmente, o elemento monitorado possui um agente que é capaz de definir as configurações de intervalo e enviar as métricas com mais informações para o gerenciador.

Armazenamento: Qual método para armazenar os dados?

Nos sistemas para monitoramento de ambientes amplamente distribuídos, o armazenamento dos dados em um Time Series DataBase (TSDB ou Banco de Série Temporal) é muito importante, pelo fato da capacidade de escalonamento e seu desempenho direcionado para sistemas de monitoramento deste tipo de aplicação. Esta é uma alternativa mais vantajosa em comparação ao uso de bancos de dados relacionais, usados por aplicações de monitoramento

tradicionais. Os TSDBs, que tem a capacidade de lidar com altas cargas de gravação e consulta, ideal para grandes quantidades de métricas. Seguindo estes parâmetros, alguns bancos de dados são recomendados para este fim como: Prometheus, TimescaleDB, OpenTSDB, Graphite, Apache Cassandra, InfluxDB e Elasticsearch. Os bancos de dados citados possuem suas próprias particularidades, métodos de consultas e interface. Ao aplicar um desses, pode elevar o tempo para aplicar, pelo fato de estudo da ferramenta.

Apresentação: Qual método de exibir os dados?

Ao coletar os dados, a apresentação é o método para visualização dos mesmos. A visualização desses dados permite uma análise do histórico e da situação atual do monitoramento, tanto de ativos da rede como de aplicações. Esta análise dos dados via painel gráfico, conhecido como *dashboard*, é essencial para identificar padrões de possíveis problemas e incidentes, de casos isolados ou até mesmo de um conjunto. Um exemplo de análise é a identificação de latência em algum aplicativo, verificando se a sobrecarga é derivada de um problema na infraestrutura ou, até mesmo, causada por alguma consulta mal elaborada internamente dentro de uma aplicação. Inúmeros sistemas usam o método de interface de painel para apresentar a saúde do ambiente, pois é uma forma mais rápida para tal análise.

Estes painéis geralmente incluem gráficos de várias formas, tabelas, listas, mapas de rede e/ou aplicações, estatísticas individuais, dentre outras. Nestes painéis, são implementadas diferentes métricas e casos de uso, sendo que a maioria costuma exibir *uptime* de serviços e as estatísticas críticas.

Alertar: Quais métodos de notificar ou sinalizar um incidente e/ou alerta?

Quando uma métrica atinge um nível acima de um limite, podem ocorrer problemas com os dispositivos e as aplicações serem afetadas, levando a falhas ou indisponibilidade do serviço. Logo, o responsável pelo sistema ou do ativo monitorado deve ser notificado sobre o evento para poder fazer as alterações necessárias, com a prioridade adequada a cada situação. Os sistemas de alerta devem ser simples de configurar. E possuir preferencialmente diferentes meios de envio das notificações, por exemplo: e-mail, mensagem para celular, ou alarme sonoro e visual em uma central de operações da rede. Ao notificar para os usuários, a mensagem precisa conter as informações necessárias para uma ágil atuação.

5.2 Arquitetura Proposta

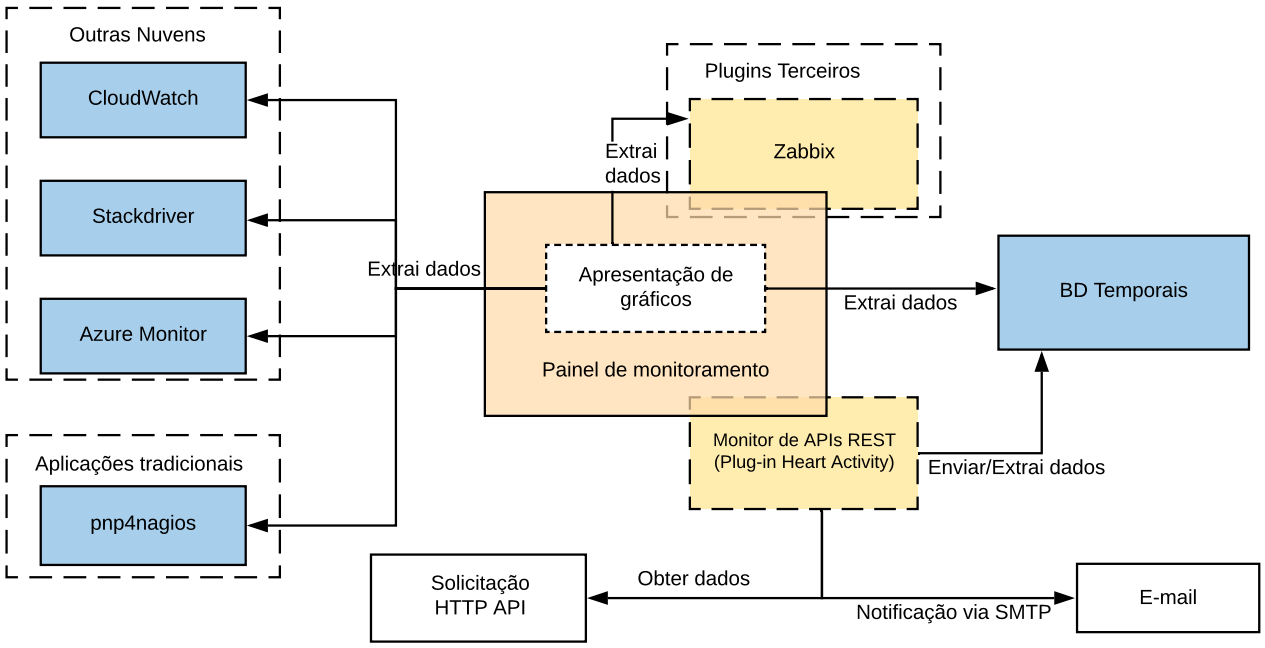
Este trabalho propõe uma arquitetura de monitoramento que contempla os principais requisitos descritos na seção 5.1: métrica, coleta, armazenamento, apresentação e alerta. A arquitetura está ilustrada na Figura 15. As métricas ou informações de monitoramento são obtidas pelos *endpoints* com padrão de resposta API RESTful, através da coleta por protocolo

HTTP/HTTPS. As principais métricas ou informações de monitoramento que são buscadas em nossa arquitetura são referentes à disponibilidade de serviços com interface do tipo REST. Essas informações são obtidas pelos *endpoints* com padrão de resposta API RESTful, através da coleta por protocolo HTTP/HTTPS. Trata-se portanto de um método de coleta sem agente e baseado em *polling*. A arquitetura proposta contempla também o monitoramento de métricas referentes a infraestruturas de nuvens públicas e privadas, assim como de infraestruturas tradicionais, com o auxílio de módulos e plug-ins de terceiros.

Os dados serão armazenados em um banco de séries temporais (e.g., InfluxDB), que pode ser implementado tanto em um contêiner como em um serviço na nuvem pública ou até em um servidor *on-premise*. É um tipo de banco que lida muito bem com informações em tempo real, com alta velocidade, causando pouco impacto no sistema. Seu tipo de sintaxe é muito parecido com a do SQL (Structured Query Language), facilitando a aplicação no cenário. Os dados armazenados serão consultados para serem exibidos por meio de gráficos em um painel de monitoramento, possibilitado por ferramentas como o Grafana, que possuem uma fácil aceitação para diversos *plug-ins* de monitoramento. Para notificar ou sinalizar um alerta, será utilizado o protocolo SMTP (Simple Mail Transfer Protocol), que permite o envio de e-mails aos administradores ou analistas de sistemas responsáveis pelo serviço monitorado. Os erros não serão tratados, somente sinalizados. Para posteriores trabalhos, os erros podem ser segmentados de melhor forma para análise e possível tratamento dos mesmos.

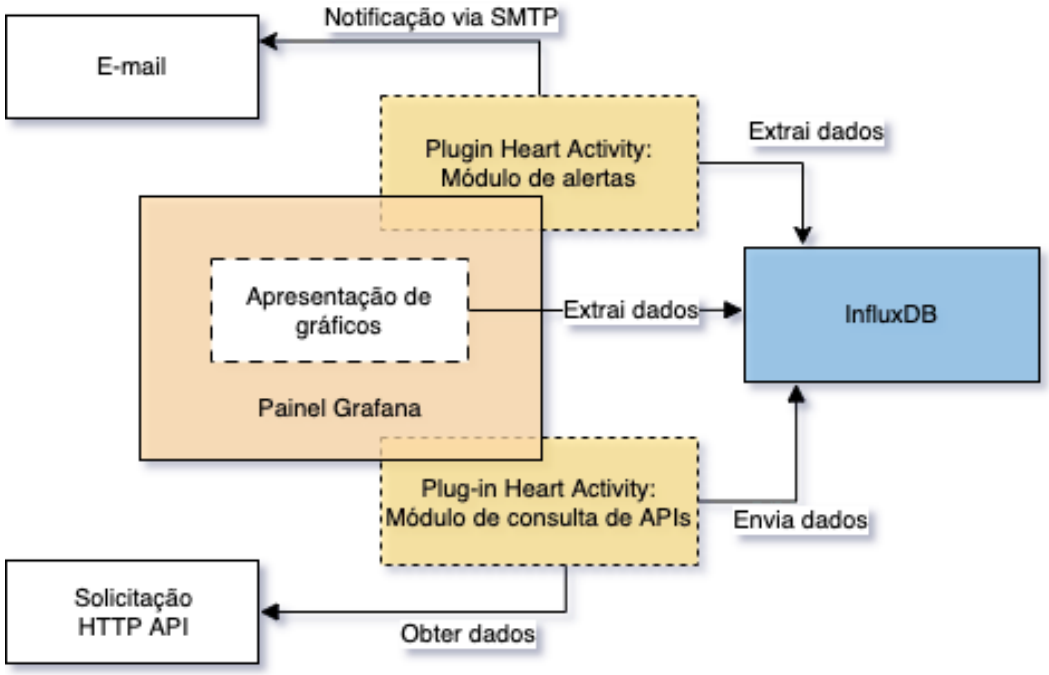
Na Figura 14 apresenta, dentre os vários componentes da arquitetura proposta, o monitor de APIs REST, que trata-se de um componente que foi instanciado neste trabalho de mestrado na forma de um *plug-in* desenvolvido e acoplado ao software do painel de monitoramento. Esse módulo é uma das contribuições originais desta dissertação, que envia requisições utilizando os protocolos HTTP (Hyper Text Transfer Protocol) e HTTPS (Hyper Text Transfer Protocol Secure) para consultar o *status code* de qualquer serviço REST. O dado obtido como resposta é gravado no banco de dados. Caso a resposta não seja 200 (OK - *Success status code*), será enviada a notificação via *e-mail* indicando o *endpoint* específico, qual protocolo falhou (HTTP ou HTTPS), o *status code* recebido na requisição que falhou e data/hora. A Figura 14 explica o fluxo do monitoramento e das requisições internas. A partir de um método GET de solicitação à API específica que esteja cadastrada, os dados serão enviados para o banco de dados, que podem ser extraídos para apresentar as informações em gráficos e alertas por *e-mail*.

Figura 14 – Representação do modelo de arquitetura da proposta



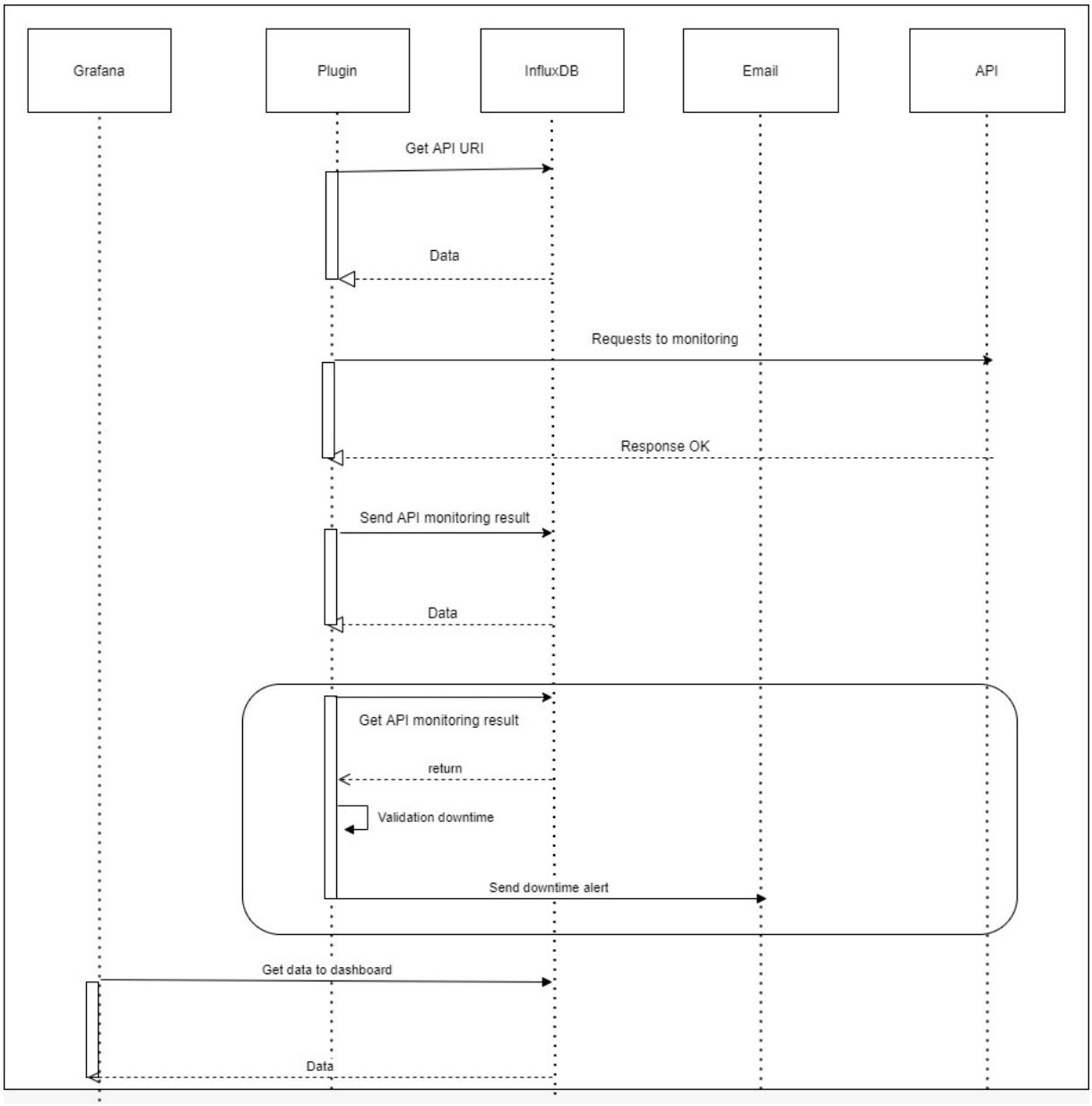
A arquitetura demonstrada acima mostra a abrangência do uso da arquitetura, que pode ser aplicada em qualquer nuvem. A proposta usa, como base, um painel de monitoramento (e.g., Grafana) que se integra a diversos Plug-ins oficiais e desenvolvidos pela comunidade, e também tem a possibilidade de se integrar com fontes de dados, tais como: (AMAZON. . . ,), GCP (STACKDRIVER, 2012), (AZURE,), (PNP4NAGIOS,), (PROMETHEUS, 2012), (DYNATRACE, 2005), (MONASCA,), (PRTG. . . , 2003) e entre outros. Além desses plug-ins oficiais, a viabilidade de desenvolver um plug-in para determinadas necessidades torna a investida valiosa. Alguns plug-ins oficiais são pagos, porém existem aplicativos desenvolvidos pela comunidade que podem agregar. A arquitetura proposta possui um complemento para o Nagios, que analisa os dados de desempenho e apresenta integração no Grafana, que é a função do pnp4nagios, que possui funções em Javascript.

Figura 15 – Representação de instanciação da arquitetura com o *plug-in Heart Activity*, Grafana *Dashboard* e banco de dados InfluxDB.



A arquitetura proposta foi implementada para testes em um sistema real. Um plug-in chamado *Heart Activity* para a ferramenta Grafana foi desenvolvido em Javascript, utilizando como ambiente de *runtime* o NodeJS. O Grafana é personalizável, flexível e possui diversas integrações com outras soluções. Por oferecer um modelo de consulta por variáveis, aumenta a sua usabilidade e dimensionamento. Usamos, como ferramenta de base de dados, InfluxDB. Neste banco guardaremos o conteúdo de fácil acesso a criações de outras métricas usadas no Grafana. A regra para composição do ciclo de verificação das APIs cadastradas é baseada na quantidade de tempo de cada verificação, podendo ser parametrizada no plug-in. Então, em uma quantidade de segundos a ser definida, o módulo de consulta de APIs realiza uma requisição HTTP/HTTPS para o endpoint cadastrado, obtendo os dados de resposta e, então, salvando em tabelas dentro da base de dados. Em paralelo, o módulo de alertas do plug-in realiza ciclos de observação dos status de resposta gravados no banco de séries temporais. Dependendo dos resultados obtidos, o módulo de alerta sinaliza por *e-mail* se alguns dos *endpoints* cadastrados estão indisponíveis, e quais as causas baseado no *status code* que é retornado do mesmo. No diagrama 16 o fluxo do *plug-in* é especificado, desenhado a sequência das requisições para monitorar as APIs.

Figura 16 – Diagrama de sequência do fluxo do *plug-in*

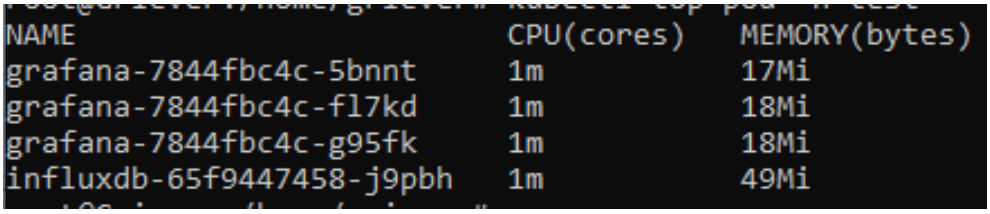


5.3 Avaliação Experimental

A arquitetura proposta nesta dissertação foi testada por meio de implementações em laboratório, com contêineres em um *cluster* alocado em um provedor de nuvem pública. Dois contêineres foram utilizados para instalação e execução do sistema de monitoramento: um para o banco InfluxDB e outro para o Grafana com o *plug-in Heart Activity*. O motivo para o uso de contêineres é uma melhor escalabilidade do sistema de monitoramento. O uso de contêineres também facilita a portabilidade para qualquer outro ambiente, pois será um pacote de software leve, autônomo e executável, que inclui tudo o que é necessário para executar um aplicativo. O Grafana funciona bem com réplicas de contêineres com um balanceador de carga, neste caso aumentando a escalabilidade. A figura 17 ilustra um *pod* de cada serviço. O *pod* é um tipo de

instância de processamento no sistema de orquestração de contêineres (KUBERNETES, 2014). Um *pod* pode conter vários contêineres. Na Figura 17 somente há um *pod* de serviço do InfluxDB. No inicio dos experimentos utilizamos somente 1 contêiner para o Grafana, analisamos que a partir de 8000 requisições, não era suficiente, então aumentamos para 3 contêineres para o Grafana, que chamamos de réplicas, e podem ser dimensionadas de acordo com a necessidade.

Figura 17 – Status pods em Kubernetes GKE



5.3.1 Google Kubernetes Engine

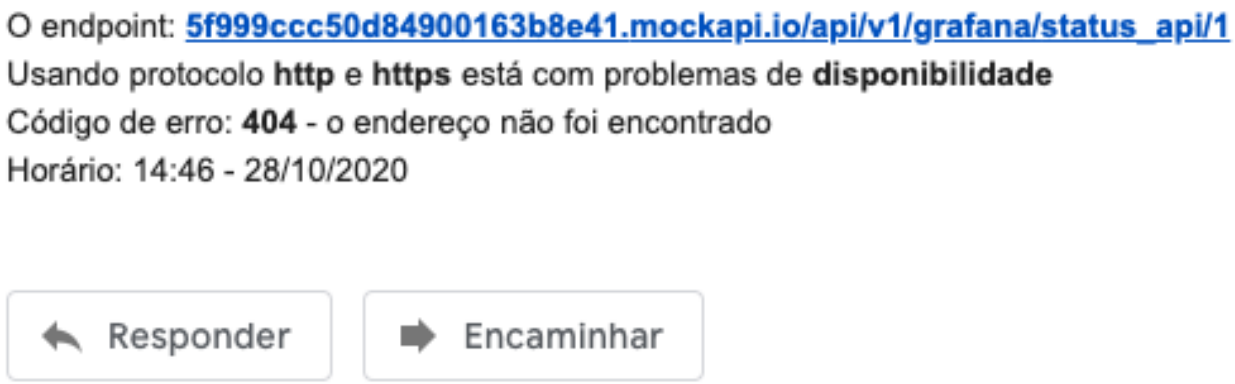
O primeiro teste realizado foi no provedor de nuvem Google Cloud, como escolhemos o orquestrador Kubernetes para gestão dos contêineres, este provedor público oferece um serviço de fácil gestão e monitoramento incluso, com análise precisa, estes são alguns dos motivos do uso nos experimentos.

Nosso primeiro experimento consiste em um *cluster* Kubernetes em um provedor de nuvem Google Cloud. Possui 3 nós com 940 mCPU *(Minimum CPU)* com 663.1 MB de memória cada. Nos primeiros testes de requisição, esse grupo de nós foi suficiente, até 8000 requisições. Para continuar acima de 8000 requisições, aumentamos mais um nó, sendo que neste a configuração foi 940 mCPU e 1,21GB de memória. O *cluster* no total possui 4 nós, totalizando 7 vCPUs e 4,70GB de memória e o disco de 40GB. O Kubernetes utilizado está na versão 1.16.13-gke.401 e os pools de nós têm a imagem Container-Optimized OS (cos), pertencente à própria Google. O Grafana e o InfluxDB estão sendo executados em contêineres, como mostrado na figura 17.

A experimentação foi feita da seguinte forma, buscando resultados do monitoramento por HTTP/HTTPS: foi criada uma API simples, que retorna, após uma requisição do tipo GET, um status *code* de confirmação 200. Essa API REST foi disponibilizada por meio de um serviço gratuito chamado (MOCKAPI, 2003). Assim, após a publicação da API, cadastramos no *plug-in Heart Activity*: (1) a URL do endpoint criado; (2) qual seria o status code que indicaria que a API estaria disponível; e (3) qual seria o intervalo para o ciclo de monitoramento (1 minuto, nesse caso). Então, para experimento, monitoramos o tempo gasto para que o módulo de verificação de API do Heart Activity fizesse a pesquisa do *endpoint* cadastrado e registrasse a ação no banco de dados (influxDB), durante 5 minutos. Em seguida, removemos a API para que em sua consulta fosse retornado um status code diferente do esperado e assim criasse uma indisponibilidade da mesma, sendo executado o procedimento de alerta. O módulo de notificação do *Heart Activity* faz, então, a leitura dos dados das APIs cadastradas, coletando o que é esperado ou não e alertando

via *e-mail*. Em caso de indisponibilidade da API, um e-mail será enviado, como no caso da Figura 18.

Figura 18 – Alerta enviado por e-mail



Na Figura 19, temos o status do monitoramento de um *endpoint*, com o período de indisponibilidade da API representado pelas barras na cor vermelha, entre os horários (14:46 a 14:59).

Figura 19 – Monitoramento do *endpoint*

Para validar a escalabilidade do ambiente de monitoramento, elevamos gradualmente a quantidade de endpoints monitorados, como um teste de estresse da ferramenta, enquanto medíamos o uso de memória e utilização de CPU do pod que hospeda o Grafana com o Heart Activity. O monitoramento foi realizado pelo período de 1 hora. A cada 1 minuto, no primeiro momento, 2000 *endpoints* foram monitorados. Posteriormente, de 4000, 6000, 8000 até 15000 *endpoints* foram monitorados por minuto.

O tempo gasto em cada execução dos experimentos é exibido na Tabela 12, assim como os dados coletados para cada um quanto ao uso de memória e utilização de CPU. Considerando que realizamos um primeiro teste de até 8000 requisições com um *pod* que possuía um único contêiner,

encontramos instabilidades e o teste foi refeito com 3 réplicas, não sendo mais necessário ativar o escalonamento para mais contêineres no teste. A Figura 20 possui os dados do uso de memória do contêiner obtidos durante este experimento de escalabilidade. Nota-se que o uso de memória manteve-se quase estável - ao redor de 59 MiB - nos experimentos entre 2000 e 10000 requisições, porém, ao aumentar a quantidade de requisições para 11000, houve um incremento de uso de memória de aproximadamente 17 MiB, chegando portanto a 76 MiB. Mesmo assim, pode-se considerar que esses aumentos não afetam significativamente a escalabilidade do sistema. Houve duas variações no ambiente: nas requisições de 8000, uma das réplicas reiniciou, porém não houveram alterações na escalabilidade, pois as outras duas réplicas se mantiveram estáveis; quando as requisições chegaram a 15000, as 3 réplicas reiniciaram, havendo um *downtime* de segundos até obter a resposta das requisições.

Figura 20 – Quantidade de requisições por minuto X uso de memória no GKE

Na Tabela 12, nota-se a coleta dos dados dos testes de requisições feitas e monitoradas pelo sistema. Estes dados foram gerados com base em toda coleta de cada requisição, gerando, assim, uma média. Analisamos o comportamento do Cluster perante o volume de requisições, que foram aumentando de acordo com a duração da carga. Nos dados da tabela 12 estão a média com quantidade de requisições por minuto, duração entre as cargas de trabalho, o uso de memória e utilização de CPU do pod. Esses valores de utilização de CPU é representado dentro do intervalo de 0,0 a 1,0, porém eventualmente o valor de utilização poderia exceder 1,0, por tratar-se de uma infraestrutura virtualizada que usa margens de segurança na alocação de recursos em casos de *overhead*.

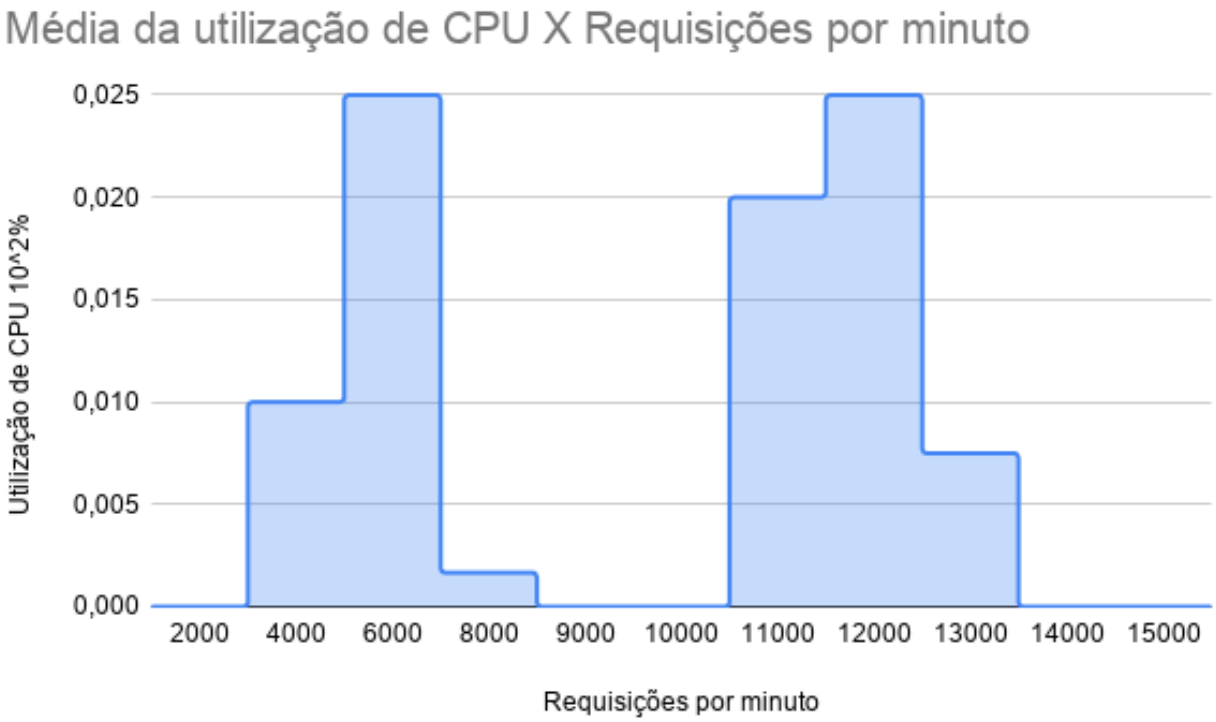
Como o teste foi efetuado em um ambiente virtualizado em Cluster, existe uma influência para o ambiente ser razoavelmente estável, com a utilização de réplicas para o Grafana e o *plug-in*, facilita o balanceamento da carga entre os pods. Devido ao balanceamento as oscilações foram pequenas. Na sigla MiB significa *MebiByte*, que é a unidade de medida para armazenamento utilizada pela Google, no caso do Cluster Kubernetes. O tempo para teste foi aumentando gradativamente de acordo com o uso de memória, para uma análise maior.

Tabela 12 – Média das Requisições por minuto e uso de CPU e memória no GKE

Requisições por minuto	**Uso de memória (MiB)**	**Uso de CPU**	**Minutos dos testes**
2000	54,4	0	5
4000	54,56	0,01	3
6000	54,25	0,025	4
8000	54,56	0	6
9000	60,25	0	8
10000	54,6	0	3
11000	76,7	0,02	2
12000	71,45	0,025	4
13000	55,7	0	4
14000	58,15	0	7
15000	60,64	0	5

A Figura 21 mostra os resultados da análise de utilização de CPU, que seguiu o mesmo parâmetro de tempo e número de *endpoints* para coletar as informações sobre uso de memória. Não houveram impactos relevantes na utilização de CPU, pois, como é possível verificar na Figura 21, em todo momento o valor se manteve estável. Usando, apenas, a análise de uso de CPU não teria a necessidade de aumentar o número de contêineres no *pod* do *cluster*. O gráfico mostra a média do uso de CPU, no qual é calculado pelo agrupamento do uso da CPU para todos os pods gerenciados atualmente pelo controlador do Cluster. As métricas de utilização de CPU é somente pelos contêineres, não incluem a utilização de demais recursos do orquestrador.

Figura 21 – Média de requisições por minuto vs Utilização de CPU no GKE



Em todo momento do teste, analisamos os dados persistidos na base de dados do InfluxDB, onde o mesmo se manteve estável mesmo com a alta demanda de dados inseridos. Como o InfluxDB é um banco direcionado para este propósito, não houve necessidade de aumentar recursos disponibilizados para o mesmo. Identificamos que, ao estressar o uso do monitoramento, houve somente aumento gradativo do uso de memória, que foi prontamente balanceado pelos nodes do *cluster*, conforme a Figura 22.

Figura 22 – Recursos dos nodes do cluster GKE

NAME	CPU(cores)	CPU%	MEMORY(bytes)	MEMORY%
gke-cluster-heart-default-pool-b7157d3d-b826	57m	6%	592Mi	93%
gke-cluster-heart-default-pool-b7157d3d-d8sz	122m	12%	662Mi	104%
gke-cluster-heart-default-pool-b7157d3d-px4l	139m	14%	637Mi	100%
gke-cluster-heart-pool-1-037121fb-skv5	628m	66%	642Mi	55%

De forma complementar realizamos mais um teste de estresse no ambiente Google (GKE) descrito na Tabela 13, o número de requisições foram iniciadas com 15000 até 20000, observamos que não foi necessário aumentar o numero de Pods do Grafana, já que o balanceamento da carga estabilizou o uso dos mesmos, ocorrendo somente reinicialização de um contêiner com requisição de 18000.

Tabela 13 – Média das Requisições por minuto e uso de CPU e memória no GKE

Requisições por minuto	Uso de memória (MiB)	Uso de CPU	Minutos dos testes
15000	77,64	0	7
16000	62,9	0	8
17000	62,66	0,01	3
18000	81,3	0,02	4
19000	73,42	0,025	4
20000	73,5	0,03	5

5.3.2 Azure Kubernetes Service

O segundo teste realizado foi no provedor de nuvem Azure da Microsoft, este provedor público oferece um serviço de fácil usabilidade no portal, com excelente documentação sobre o serviço, estes são alguns dos motivos do uso nos experimentos.

O experimento consiste em um *cluster* Kubernetes em um provedor de nuvem Azure, no qual chamamos de AKS (Serviço de Kubernetes do Azure). Possui 2 nós com 2 vCPU *(Minimum CPU)* com 7GB de memória cada. Em todo momento dos testes a quantidade dos nós foram suficientes. O *cluster* no total possui 2 nós, totalizando 4 vCPUs e 14GB de memória e o disco de 40GB. O Kubernetes utilizado está na versão 1.18.14 e os pools de nós têm a imagem Linux. O Grafana e o InfluxDB estão sendo executados em contêineres, como mostrado na Figura 23.

Figura 23 – Status Pods AKS

```
NAME                          CPU(cores)   MEMORY(bytes)
grafana-58667b54b9-jmjb6      1m           22Mi
grafana-58667b54b9-q6vc5      1m           24Mi
grafana-58667b54b9-xrr5j      1m           23Mi
influxdb-0                    3m           13Mi
```

Analisamos o monitoramento tanto no painel AKS como por CLI (Interfaces de Linhas de Comando) utilizando o kubectl. O kubectl é uma ferramenta para linha de comando que permite um melhor gerenciamento dos recursos, trazendo uma análise mais completa. Utilizamos o painel AKS para verificar seu próprio monitoramento, tanto do Cluster, como dos nós e contêineres, nas Figuras 24 e 25 tem a porcentagem de utilização das últimas 6 horas do Cluster, com suas taxas de média e máximo de uso de CPU e memória.

Figura 24 – Utilização de CPU Cluster AKS

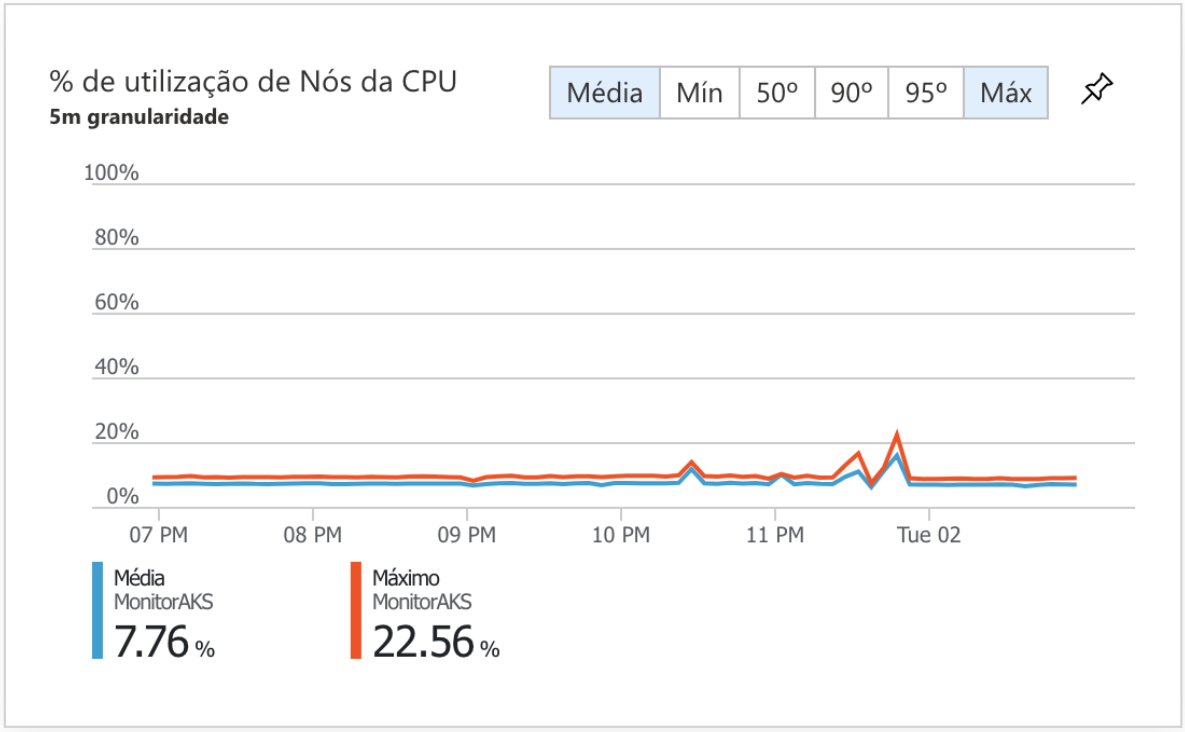
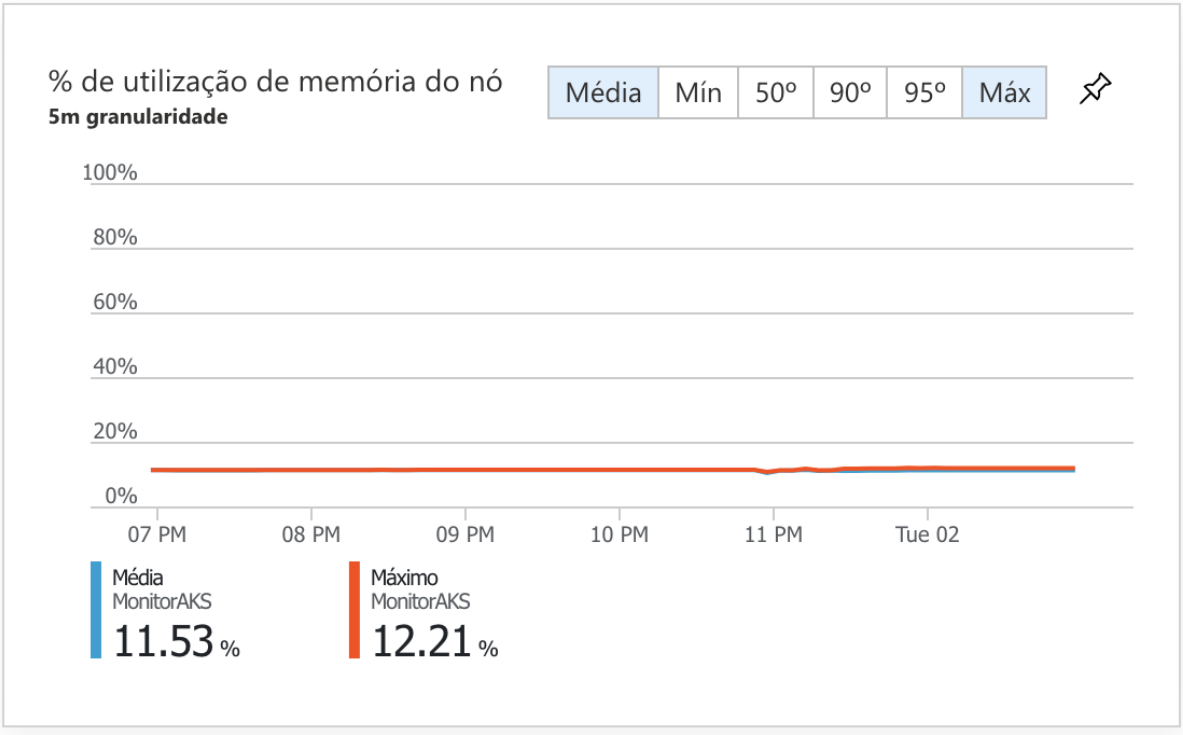


Figura 25 – Utilização de memória Cluster AKS



A visualização da utilização dos recursos via kubectl mostra a unidade de CPU em cores e de memória em bytes, e as porcentagens da capacidade alocável do nó que esses números

representam. Como esboçado na Figura 26 na infraestrutura Azure e na Figura 22 da Google.

Figura 26 – Recursos dos nodes do cluster AKS

```
NAME                                CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
aks-nodepool1-32050393-vmss000000   170m         8%     1841Mi          40%
aks-nodepool1-32050393-vmss000001   88m          4%     1759Mi          38%
```

Iniciamos os testes com 3 replicas do Grafana, e efetuamos as requisições seguindo os mesmos padrões dos testes realizados no GKE (Google Kubernetes Engine), porém com os valores para testes de 2000 até 20000 em um único experimento, analisamos o comportamento dos pods também no painel, como na Figura 27, neste possui a média de uso da última hora da CPU em mili-núcleos em suas métricas, na Figura 28 possui a média do conjunto de trabalho da memória.

Figura 27 – Utilização de CPU contêiner Grafana

NOME	STATUS	% D...	MÉDIA	POD	NÓ	REINICIA	TEMPO...
grafana	Ok	0%	0.6 mc	grafana-586...	aks-nodepo...	0	2 horas
grafana	Ok	0%	0.6 mc	grafana-586...	aks-nodepo...	0	2 horas
grafana	Ok	0%	0.5 mc	grafana-586...	aks-nodepo...	0	2 horas

Figura 28 – Utilização de memória contêiner Grafana

NOME	STATUS	% DE TEMP...	MÉDIA	POD	NÓ	REINICIA	TEMPO DE A...
grafana	Ok	0.5%	24.91 MB	grafana-58667b54b...	aks-nodepool1-320...	0	2 horas
grafana	Ok	0.5%	23.67 MB	grafana-58667b54b...	aks-nodepool1-320...	0	2 horas
grafana	Ok	0.5%	23.45 MB	grafana-58667b54b...	aks-nodepool1-320...	0	2 horas

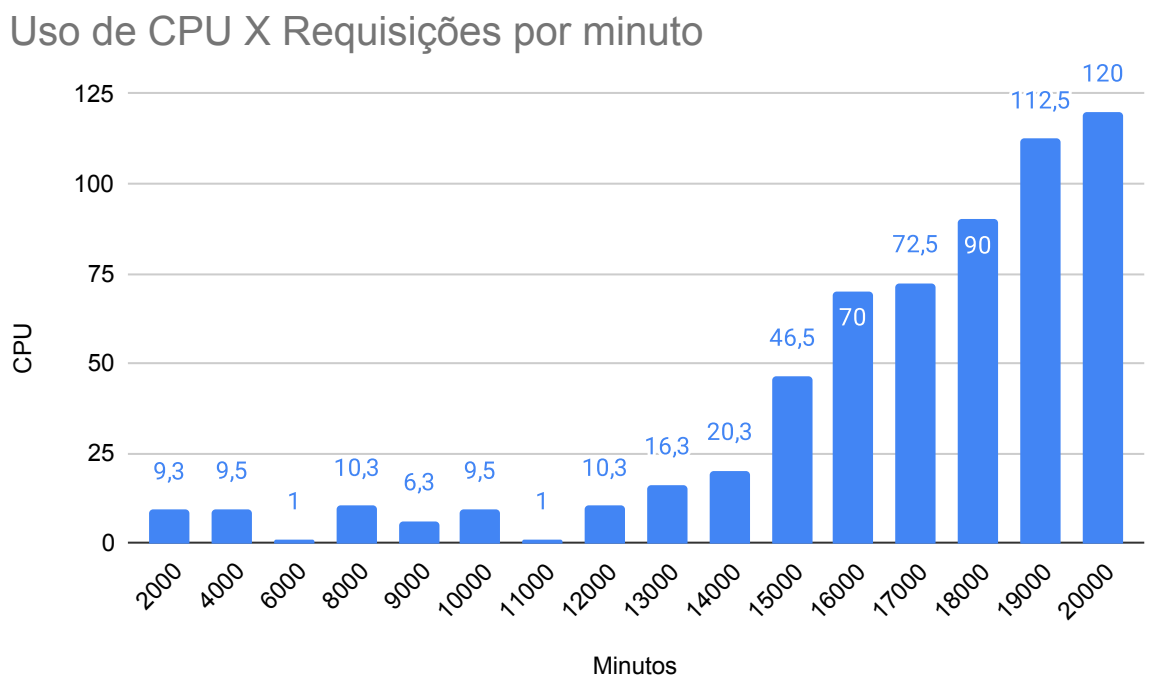
A Figura 30 possui os dados do uso de memória/CPU dos contêineres obtidos durante este experimento de escalabilidade. Observa-se que o uso de memória manteve-se quase estável - ao redor de 23 MiB - nos experimentos entre 2000 e 13000 requisições, porém, ao aumentar a quantidade de requisições para 14000, houve um aumento irrisório de 1 MiB, chegando portanto a 24 MiB. Consideramos que esses aumentos não afetam significativamente a escalabilidade do sistema em nada. Houve três variações no ambiente: nas requisições de 11000, duas das réplicas reiniciaram, porém não houveram alterações na escalabilidade, pois a outra réplica se mantivera estável; quando as requisições chegaram a 14000, uma replica reiniciou, em nível de uso de memória foi baixo; e no penúltimo teste de 19000 algumas requisições foram perdidas por 1 minuto, logo estabeleceu. Todos os dados são especificados em média na Tabela 14

Tabela 14 – Média das Requisições por minuto e uso de CPU e memória no Cluster AKS

Requisições por minuto	Uso de memória (MiB)	Uso de CPU	Minutos dos testes
2000	23	9,3	5
4000	23	9,5	4
6000	23	1	3
8000	23	10,3	6
9000	23,33	6,3	9
10000	23	9,5	3
11000	23,33	1	2
12000	23	10,3	5
13000	23,6	16,3	5
14000	24	20,3	9
15000	24	46,5	6
16000	23	70	8
17000	24	72,5	2
18000	24,6	90	3
19000	25	112,5	4
20000	25	120	5

O uso de CPU variou mais, em relação a memória, a média geral foi de 37,83 de uso de CPU, considerando a estabilidade que o sistema se manteve, foram ótimos resultados, e com um crescimento gradativo.

Figura 29 – Média de utilização de CPU AKS



A média do intervalo de tempo entre os testes foi de 4,93, os testes foram observados, a

medida que aumentava ou estabelecia o uso dos recursos do cluster. No experimento analisamos os dados persistidos na base de dados do InfluxDB, onde o mesmo se manteve estável mesmo com a alta demanda de dados inseridos.

Esta saída das Figuras 30 e 31 mostra três pods de trabalho (Grafana) e um pod (InfluxDB) em um cluster do AKS. Cada linha exibe a quantidade total de CPU (em núcleos ou, neste caso, m por milicores) e memória (em MiB) que o pod está usando.

Figura 30 – Utilização de CPU e memória Pods AKS

```
NAME                       CPU(cores)   MEMORY(bytes)
grafana-58667b54b9-jmjb6   9m           22Mi
grafana-58667b54b9-q6vc5   3m           24Mi
grafana-58667b54b9-xrr5j   16m          23Mi
influxdb-0                 1m           13Mi
```

Na Figura 30 mostra como os pods mantiveram seus recursos por mais tempo, na maior parte dos testes, sempre se estabelecia o uso de memória e CPU. Apenas nos últimos testes que houve um aumento significante de uso de CPU, como o demonstrado na Figura 31.

Figura 31 – Utilização de CPU e memória Pods AKS em alta

```
NAME                       CPU(cores)   MEMORY(bytes)
grafana-58667b54b9-jmjb6   126m         22Mi
grafana-58667b54b9-q6vc5   120m         25Mi
grafana-58667b54b9-xrr5j   114m         24Mi
influxdb-0                 2m           13Mi
```

5.4 Considerações

Os experimentos realizados nos dois provedores de nuvem pública mostrou resultados distintos, enquanto o Google obteve uso maior dos recursos da memória, o Cluster da Azure obteve mais da CPU, porém de forma geral os dois mantiveram os sistemas disponíveis, e em nenhum dos provedores houveram dificuldades para implementar os testes.

Para trabalhos futuros poderia ser realizado estes mesmos experimentos em uma nuvem privada, como o OpenStack, em projetos pilotos executamos e houveram ótimos resultados, porém a desvantagem do uso na nuvem privada, é para criar o cluster, pois demanda mais tempo, os demais fatores prosseguem iguais os das nuvens públicas.

6
Conclusão

As soluções atuais para monitoramento de código aberto e proprietárias oferecem suporte a muitas funcionalidades e métricas. No entanto, elas são muito complexas, com uma curva de aprendizado acentuada, e muitas são baseadas em agentes. A prova de conceito apresentada aqui mostra uma abordagem de monitoramento com boa relação custo-benefício, com baixa necessidade de habilidades avançadas para sua configuração e utilização. Com a facilidade do painel de apresentação (e.g., InfluxDB), existem *Plug-ins* oficiais e os desenvolvidos pela comunidade, sendo permitido customizar o monitoramento para outros fins. Como desenvolvemos um *plug-in* para cadastrar os *endpoints* das aplicações e monitorar as APIs, pode ser colocado em prática de forma simples. Essa abordagem seria atraente para diversos fins, dentre eles a aplicação em organizações com necessidades mais simples e orçamentos pequenos, já que podem ser implementados em uma infraestrutura computacional pré-existente. Essa abordagem possui boa escalabilidade, pois, ao adicionar ou remover réplicas de contêineres, oferece suporte a aumentar o número de APIs a serem monitoradas. Além disso, essa abordagem pode ser usada em um ambiente de nuvem com facilidade para visualização dos dados coletados e integração com outros sistemas.

Nas seções seguintes, apresentamos as considerações finais, as principais contribuições e os trabalhos futuros.

6.1 Considerações finais

Neste trabalho foi realizada uma revisão bibliográfica sobre as arquiteturas, sistemas e aplicações para monitoramento em computação de nuvens. A revisão foi realizada por meio de uma revisão sistemática da literatura, que respondeu algumas questões de pesquisa com o objetivo de traçar um panorama do tema. Por meio da revisão, foram identificadas as principais aplicações para monitoramento de infraestruturas híbridas, os tipos de arquitetura de monitoramento e as

métricas monitoradas.

A partir da leitura dos trabalhos relacionados obtidos através da revisão, foi verificado que havia uma lacuna em monitoramento para aplicações no padrão API REST. Não foram encontrados trabalhos que abordassem o monitoramento dos aplicativos hospedados na nuvem. Essa visão de monitoramento dessas aplicações alocadas foi considerada um dos objetos de estudo, haja vista o crescimento de aplicações em padrão API REST e da arquitetura de microsserviços, que comumente utiliza este padrão. Pela necessidade de um monitoramento para este fim, demandou-se o desenvolvimento de um *plug-in* a ser disponibilizado na arquitetura proposta.

Um dos principais desafios para o monitoramento de infraestruturas virtuais é a alocação dos registros de todas ferramentas que são utilizadas em um lugar centralizado. Cada vez mais, a computação em nuvem se firma presente no cotidiano, não apenas para empresas, mas também na área acadêmica. A utilização de um conjunto de ferramentas para centralização de eventos presentes nos sistemas tendem a ser utilizados pela sua eficiência em dinamizar as integrações de vários programas a ajudar a monitorar, de forma intensiva, a arquitetura em nuvem. A implementação desta arquitetura de monitoramento para infraestruturas de computação fornecerá um suporte maior para a comunidade para a colhida de informações e registros.

6.2 Principais Contribuições

Resumidamente, as principais contribuições desse trabalho são descritas:

- Um plug-in para monitoramento da disponibilidade de aplicações com padrões RESTful;
- Uma arquitetura de monitoramento integrável com ferramentas, outras nuvens e aplicações diversas;
- A avaliação da escalabilidade da arquitetura e do funcionamento do *plug-in* perante a testes por requisições.

Por meio da avaliação experimental, pode-se afirmar que as contribuições propostas foram alcançadas. Os resultados experimentais demonstraram que a arquitetura suporta adequadamente as requisições e podem ser dimensionáveis, aumentando e diminuindo conforme necessário, garantindo redução de custos. Por sua vez, o modelo da arquitetura proposta é útil para a integração com outras nuvens (públicas e privadas), com outras aplicações tradicionais como o Nagios.

6.3 Trabalhos Futuros

Para referência de trabalhos futuros, o sistema proposto deve contemplar semáforos (controle de distribuição) para atomizar o sincronismo balanceado das requisições em cada

contêiner, dividindo a quantidade de requisições de modo assíncrono entre a quantidade de contêineres disponíveis por igual. Esse tipo de configuração contribuiria para avaliar sua robustez e escalabilidade em termos de balanceamento de carga.

Outro caminho promissor é estender a arquitetura de monitoramento aqui proposta para permitir o monitoramento de informações de desempenho, tais como tempo de resposta das APIs e uma maior variedade de métodos dos protocolos HTTP/HTTPS ou, inclusive, outros protocolos.

Referências

ACCELOPS. 2016. <http://www.accelops.com/>. Acessado: 2019-08-14. Citado na página 16.

ACETO, G. et al. Cloud monitoring: Definitions, issues and future directions. In: *2012 1st IEEE International Conference on Cloud Networking, CLOUDNET 2012 - Proceedings*. [S.l.: s.n.], 2012. p. 63–67. ISBN 9781467327985. Citado 2 vezes nas páginas 50 e 59.

ACETO, G. et al. Cloud monitoring: A survey. *Computer Networks*, Elsevier B.V., v. 57, n. 9, p. 2093–2115, 2013. ISSN 13891286. Disponível em: <http://dx.doi.org/10.1016/j.comnet.2013.04.001>. Citado na página 50.

AKTAS, M. S. Hybrid cloud computing monitoring software architecture. *Concurrency Computation*, v. 30, n. 21, p. 1–9, 2018. ISSN 15320634. Citado 6 vezes nas páginas 15, 24, 45, 46, 53 e 57.

AKTAS, M. S. Hybrid cloud computing monitoring software architecture. *Concurrency Computation*, v. 30, n. 21, p. 1–9, 2018. ISSN 15320634. Citado 2 vezes nas páginas 34 e 47.

AL-SHAMMARI, S. W.; HUSEIN, A. A. Response Time Study of Cloud Web Application - Based Smart Monitoring System. *Proceedings of the 2020 International Conference on Computer Science and Software Engineering, CSASE 2020*, p. 138–141, 2020. Citado 3 vezes nas páginas 48, 55 e 57.

ALCARAZ CALERO, J. M.; AGUADO, J. G. MonPaaS: An adaptive monitoring platformas a service for cloud computing infrastructures and services. *IEEE Transactions on Services Computing*, v. 8, n. 1, p. 65–78, 2015. ISSN 19391374. Citado 9 vezes nas páginas 16, 17, 34, 39, 45, 46, 47, 52 e 56.

ALCARAZ CALERO, J. M.; Gutiérrez Aguado, J. Comparative analysis of architectures for monitoring cloud computing infrastructures. *Future Generation Computer Systems*, Elsevier B.V., v. 47, p. 16–30, 2015. ISSN 0167739X. Disponível em: <http://dx.doi.org/10.1016/j.future.2014.12.008>. Citado 3 vezes nas páginas 15, 44 e 52.

AMAZON CloudWatch. <https://aws.amazon.com/pt/cloudwatch/>. Acessado: 2019-08-10. Citado 2 vezes nas páginas 16 e 62.

ANEKA: A Software Platform for .NET-based Cloud Computing. 2009. <https://arxiv.org/abs/0907.4622>. Acessado: 2019-08-19. Citado na página 43.

APACHE CloudStack. 2010. <https://cloudstack.apache.org/>. Acessado: 2019-08-19. Citado na página 43.

AZURE. <https://azure.microsoft.com/>. Acessado: 2019-08-10. Citado 2 vezes nas páginas 16 e 62.

BRATTSTROM, M.; MORREALE, P. Scalable Agentless Cloud Network Monitoring. *Proceedings - 4th IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2017 and 3rd IEEE International Conference of Scalable and Smart Cloud, SSC 2017*, IEEE, p. 171–176, 2017. Citado 7 vezes nas páginas 34, 45, 46, 47, 53, 56 e 59.

CARVALHO, M. B. de. *Um framework para a construção automatizada de cloud monitoring slices baseados em múltiplas soluções de monitoramento*. Tese (Dissertação (mestrado)) — Universidade Federal do Rio Grande do Sul, 2015. Citado 5 vezes nas páginas 34, 38, 44, 46 e 47.

CIUFFOLETTI, A. Design of a Cloud Monitoring System Beyond Nagios. v. 2, n. Closer, p. 363–370, 2016. Citado 8 vezes nas páginas 34, 39, 45, 46, 47, 48, 53 e 56.

CLOUDKICK. 2008. <https://www.crunchbase.com/organization/cloudkick>. Acessado: 2019-08-13. Citado na página 16.

Currie A.R., McConnell A., Parr G.P., M. S.; K., K. Truesource: A True Performance For Hierarchical Cloud Monitoring. In: . [S.l.: s.n.], 2014. p. 516–519. ISBN 978-1-4799-7881-6. Citado 4 vezes nas páginas 34, 45, 46 e 47.

DAVIS, G. 2020: Life with 50 billion connected devices. IEEE, p. 1–1, 2018. Citado na página 14.

De Chaves, S. A.; URIARTE, R. B.; WESTPHALL, C. B. Toward an architecture for monitoring private clouds. *IEEE Communications Magazine*, v. 49, n. 12, p. 130–137, 2011. ISSN 01636804. Citado 8 vezes nas páginas 23, 34, 38, 45, 46, 47, 51 e 56.

DIEGO. <https://docs.pivotal.io/pivotalcf/2-3/concepts/diego/diego-architecture.html>. Acessado: 2019-08-16. Citado na página 26.

DOCKER Swarm. <https://docs.docker.com/engine/swarm/>. Acessado: 2019-08-16. Citado na página 26.

DUAN, Y. et al. Everything as a Service (XaaS) on the Cloud: Origins, Current and Future Trends. *Proceedings - 2015 IEEE 8th International Conference on Cloud Computing, CLOUD 2015*, n. November 2017, p. 621–628, 2015. Citado na página 22.

DYNATRACE. 2005. <https://www.dynatrace.com/>. Acessado: 2019-08-21. Citado na página 62.

EMEAKAROHA, V. C. et al. Towards autonomic detection of SLA violations in Cloud infrastructures. *Future Generation Computer Systems*, Elsevier B.V., v. 28, n. 7, p. 1017–1029, 2012. ISSN 0167739X. Disponível em: <http://dx.doi.org/10.1016/j.future.2011.08.018>. Citado 4 vezes nas páginas 34, 45, 46 e 47.

EUCALYPTUS cloud. 2008. <https://www.eucalyptus.cloud/>. Acessado: 2019-08-14. Citado na página 51.

FAHAD, A. M.; AHMED, A. A.; KAHAR, M. N. M. The importance of monitoring cloud computing: An intensive review. In: *IEEE Region 10 Annual International Conference, Proceedings/TENCON*. [S.l.: s.n.], 2017. v. 2017-Decem, p. 2858–2863. ISBN 9781509011339. ISSN 21593450. Citado na página 15.

FATEMA, K. et al. A survey of Cloud monitoring tools: Taxonomy, capabilities and objectives. *Journal of Parallel and Distributed Computing*, v. 74, n. 10, p. 2918–2933, 2014. ISSN 07437315. Citado 2 vezes nas páginas 40 e 50.

FIELDING, R. *Representational State Transfer (REST)*. 2000. Acessado: 2019-08-13. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>. Citado na página 20.

FONSECA, J. J. S. *Metodologia da pesquisa científica*. [S.l.]: Fortaleza: UEC, 2002. Citado na página 18.

GANGLIA. 2016. <http://ganglia.sourceforge.net/>. Acessado: 2019-08-18. Citado na página 40.

GERHARDT T.; SILVEIRA, D. *Métodos de pesquisa.* [S.l.]: Editora da UFRGS, 2009. Citado na página 18.

GILL, A. Q.; HEVARY, S. Cloud monitoring data challenges: A systematic review. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 9947 LNCS, p. 72–79, 2016. ISSN 16113349. Citado na página 51.

GRAFANA. 2014. <https://grafana.com/>. Acessado: 2019-08-15. Citado na página 16.

GUTIERREZ-AGUADO, J.; Alcaraz Calero, J. M.; Diaz Villanueva, W. IaaSMon: Monitoring Architecture for Public Cloud Computing Data Centers. *Journal of Grid Computing*, Journal of Grid Computing, v. 14, n. 2, p. 283–297, 2016. ISSN 15729184. Disponível em: <http://dx.doi.org/10.1007/s10723-015-9357-4>. Citado 7 vezes nas páginas 34, 45, 46, 47, 52, 53 e 56.

HEAT. <https://wiki.openstack.org/wiki/Heat>. Acessado: 2019-08-15. Citado na página 26.

IBM Tivoli. <https://www.ibm.com/support/knowledgecenter/en/SSTFXA_6.3.0/com.ibm.itm.doc_6.3/install/itm_over.htm>. Acessado: 2019-08-21. Citado na página 40.

INFLUXDB. 2013. <https://www.influxdata.com/>. Acessado: 2019-08-15. Citado na página 16.

KATSAROS, G. et al. A Self-adaptive hierarchical monitoring mechanism for Clouds. *Journal of Systems and Software*, Elsevier Inc., v. 85, n. 5, p. 1029–1041, 2012. ISSN 01641212. Disponível em: <http://dx.doi.org/10.1016/j.jss.2011.11.1043>. Citado 4 vezes nas páginas 34, 45, 46 e 47.

KERNEL-BASED Virtual Machine. 2007. <https://www.linux-kvm.org/page/Main_Page>. Acessado: 2019-08-20. Citado na página 51.

KITCHENHAM, B. *Procedures for performing systematic reviews.* [S.l.]: Keele, UK, Keele University, 2004. v. 33. 1–26 p. Citado na página 28.

KUBERNETES. 2014. <https://kubernetes.io/>. Acessado: 2019-08-15. Citado 2 vezes nas páginas 26 e 65.

KUBLER, E.; MINOR, M. Towards cross-layer monitoring of cloud workflows. *CEUR Workshop Proceedings*, v. 1458, p. 288–295, 2015. ISSN 16130073. Citado 4 vezes nas páginas 34, 45, 46 e 47.

LU, X. et al. JTangCMS: An efficient monitoring system for cloud platforms. *Information Sciences*, Elsevier Inc., v. 370-371, p. 402–423, 2016. ISSN 00200255. Disponível em: <http://dx.doi.org/10.1016/j.ins.2016.06.009>. Citado 6 vezes nas páginas 34, 45, 46, 47, 54 e 57.

MARATHON. <https://mesosphere.github.io/marathon/>. Acessado: 2019-08-15. Citado na página 26.

MAURO, D.; SCHMIDT, K.; LOUKIDES, M. *Essential SNMP*. O'Reilly, 2001. (Help for system and network administrators). ISBN 9780596000202. Disponível em: <https://books.google.com.br/books?id=sT4PTu1TwHgC>. Citado na página 59.

MOCKAPI. 2003. <https://www.mockapi.io/>. Acessado: 2019-08-21. Citado na página 65.

MONASCA. <https://monasca.io/>. Acessado: 2019-08-21. Citado na página 62.

MONITIS. 2007. <https://www.monitis.com/>. Acessado: 2019-08-18. Citado na página 41.

MONTES, J. et al. GMonE: A complete approach to cloud monitoring. *Future Generation Computer Systems*, Elsevier B.V., v. 29, n. 8, p. 2026–2040, 2013. ISSN 0167739X. Disponível em: <http://dx.doi.org/10.1016/j.future.2013.02.011>. Citado 6 vezes nas páginas 34, 45, 46, 47, 54 e 57.

MORADI, F. et al. ConMon: An automated container based network performance monitoring system. *Proceedings of the IM 2017 - 2017 IFIP/IEEE International Symposium on Integrated Network and Service Management*, p. 54–62, 2017. Citado 6 vezes nas páginas 34, 45, 46, 47, 55 e 57.

MRTG. 1995. <https://www.mrtg.com/>. Acessado: 2019-08-14. Citado na página 16.

NAGIOS. 2002. <https://www.nagios.org/>. Acessado: 2019-08-13. Citado na página 16.

NATU, M. et al. Holistic performance monitoring of hybrid clouds: Complexities and future directions. *IEEE Cloud Computing*, v. 3, n. 1, p. 72–81, Jan 2016. ISSN 2325-6095. Citado na página 17.

NCONF. <https://www.nconf.org/>. Acessado: 2019-08-20. Citado na página 52.

NEUMANN, A.; Laranjeiro, N.; Bernardino, J. An analysis of public rest web service apis. *IEEE Transactions on Services Computing*, p. 1–1, 2018. ISSN 1939-1374. Citado na página 48.

NIMSOFT. 1998. <https://support.nimsoft.com/>. Acessado: 2019-08-18. Citado na página 41.

OPENNEBULA. 2008. <https://opennebula.io/>. Acessado: 2019-08-15. Citado na página 51.

OPENSTACK, Open Source Cloud Computing Infrastructure. 2010. <https://www.openstack.org/>. Acessado: 2019-08-20. Citado na página 52.

PAHL, C. et al. Cloud Container Technologies: a State-of-the-Art Review. *IEEE Transactions on Cloud Computing*, n. May, 2017. ISSN 21687161. Citado na página 26.

PEREZ-ESPINOZA, J. A. et al. A Distributed Architecture for Monitoring Private Clouds. *Proceedings - International Workshop on Database and Expert Systems Applications, DEXA*, IEEE, v. 2016-Febru, p. 186–190, 2016. ISSN 15294188. Citado 6 vezes nas páginas 34, 45, 46, 47, 51 e 56.

Peter Mell (NIST), T. G. N. The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology. *Acta Horticulturae*, v. 728, p. 269–274, 2006. ISSN 05677572. Citado na página 21.

PETERSEN, e. a. *Systematic mapping studies in software engineering*. [S.l.]: In: EASE, 2008. v. 8. 68–77 p. Citado 2 vezes nas páginas 28 e 48.

PNP4NAGIOS. <http://docs.pnp4nagios.org/start>. Acessado: 2019-08-21. Citado na página 62.

POPEK, G. J.; GOLDBERG, R. P. Formal requirements for virtualizable third generation architectures. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 17, n. 7, p. 412–421, jul. 1974. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/361011.361073>. Citado na página 44.

POVEDANO-MOLINA, J. et al. DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds. *Future Generation Computer Systems*, Elsevier B.V., v. 29, n. 8, p. 2041–2056, 2013. ISSN 0167739X. Disponível em: <http://dx.doi.org/10.1016/j.future.2013.04.022>. Citado 6 vezes nas páginas 34, 43, 46, 47, 54 e 57.

PROMETHEUS. 2012. <https://prometheus.io/>. Acessado: 2019-08-21. Citado na página 62.

PRTG Network Monitor. 2003. <https://www.paessler.com/network_monitor_software>. Acessado: 2019-08-21. Citado na página 62.

SÁ, T. T. *Uma estratégia de gerenciamento de infraestrutura de datacenters baseada em técnicas de monitoramento distribuído e controle centralizado*. Tese (Doutorado) — Universidade Federal do Ceará, 2013. Citado 5 vezes nas páginas 34, 38, 45, 46 e 47.

SAMPAIO, A. R. et al. Improving microservice-based applications with runtime placement adaptation. *Journal of Internet Services and Applications*, Journal of Internet Services and Applications, v. 10, n. 1, 2019. ISSN 18690238. Citado na página 48.

SAMPAIO R. F.; MANCINI, M. C. *Estudos de revisão sistemática: um guia para síntese criteriosa da evidência científica*. [S.l.]: Rev. bras. fisioter., São Carlos, 2007. v. 11. 83–89 p. Citado 2 vezes nas páginas 8 e 29.

SCHUBERT, F. F. d. S. C. *UMA ARQUITETURA DE COMPUTACAO AUTONOMA E COGNITIVA PARA MONITORAMENTO DE NUVENS*. 44 p. Tese (Doutorado) — Universidade Federal de Santa Catarina, 2014. Citado 5 vezes nas páginas 34, 38, 45, 46 e 47.

SMIT, M.; SIMMONS, B.; LITOIU, M. Distributed, application-level monitoring for heterogeneous clouds using stream processing. *Future Generation Computer Systems*, Elsevier B.V., v. 29, n. 8, p. 2103–2114, 2013. ISSN 0167739X. Disponível em: <http://dx.doi.org/10.1016/j.future.2013.01.009>. Citado 6 vezes nas páginas 15, 34, 39, 45, 46 e 47.

SOLTESZ, S. et al. Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. In: *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*. New York, NY, USA: Association for Computing Machinery, 2007. (EuroSys '07), p. 275–287. ISBN 9781595936363. Disponível em: <https://doi-org.ez20.periodicos.capes.gov.br/10.1145/1272996.1273025>. Citado na página 25.

STACKDRIVER. 2012. <https://cloud.google.com/products/operations>. Acessado: 2019-08-21. Citado na página 62.

STEPHEN, A.; BENEDICT, S.; KUMAR, R. P. Monitoring IaaS using various cloud monitors. *Cluster Computing*, Springer US, p. 1–13, 2018. ISSN 15737543. Disponível em: <https://doi.org/10.1007/s10586-017-1657-y>. Citado 2 vezes nas páginas 55 e 57.

SUN, Y. et al. An architecture model of management and monitoring on Cloud services resources. *ICACTE 2010 - 2010 3rd International Conference on Advanced Computer Theory and Engineering, Proceedings*, v. 3, p. 207–211, 2010. ISSN 1098-6596. Citado 4 vezes nas páginas 34, 45, 46 e 47.

SYED, H. J. et al. Cloud monitoring: A review, taxonomy, and open research issues. *Journal of Network and Computer Applications*, Elsevier Ltd, v. 98, n. March, p. 11–26, 2017. ISSN 10958592. Disponível em: <http://dx.doi.org/10.1016/j.jnca.2017.08.021>. Citado na página 40.

SYED, H. J. et al. CloudProcMon: A non-intrusive cloud monitoring framework. *IEEE Access*, IEEE, v. 6, p. 44591–44606, 2018. ISSN 21693536. Citado 6 vezes nas páginas 34, 45, 46, 47, 53 e 57.

TAHERIZADEH, S. et al. Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review. *Journal of Systems and Software*, Elsevier Inc., v. 136, p. 19–38, 2018. ISSN 01641212. Disponível em: <https://doi.org/10.1016/j.jss.2017.10.033>. Citado na página 16.

TEAMVIEWER. 2005. <https://www.teamviewer.com/>. Acessado: 2019-08-18. Citado na página 41.

XEN. <https://xen-orchestra.com/#!/xo-home>. Acessado: 2019-08-21. Citado na página 51.

XU, X.; CHEN, Y.; Alcaraz Calero, J. M. Distributed decentralized collaborative monitoring architecture for cloud infrastructures. *Cluster Computing*, Springer US, v. 20, n. 3, p. 2451–2463, 2017. ISSN 15737543. Citado 4 vezes nas páginas 34, 44, 46 e 47.

ZABBIX. *The Enterprise-Class Open Source Network Monitoring*. 2020. Acessado: 2019-08-13. Disponível em: <https://www.zabbix.com>. Citado 2 vezes nas páginas 16 e 40.

ZOU, D. et al. Design and implementation of a trusted monitoring framework for cloud platforms. *Future Generation Computer Systems*, Elsevier B.V., v. 29, n. 8, p. 2092–2102, 2013. ISSN 0167739X. Disponível em: <http://dx.doi.org/10.1016/j.future.2012.12.020>. Citado 4 vezes nas páginas 34, 44, 46 e 47.