Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Cloud.Jus 3.0: Plataforma de Gerenciamento em Nuvem para o Ambiente de Infraestrutura do Poder Judiciário da União

Gabriel Rodrigues Diógenes Macêdo

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof.a Dr.a Aletéia Patrícia Favacho de Araújo Von Paumgartten

Brasília
2023

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Cloud.Jus 3.0: Plataforma de Gerenciamento em Nuvem para o Ambiente de Infraestrutura do Poder Judiciário da União

Gabriel Rodrigues Diógenes Macêdo

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof.a Dr.a Aletéia Patrícia Favacho de Araújo Von Paumgartten (Orientador)
CIC/UnB

Prof. Dr. Flávio de Barros Vidal
CIC/UnB

Prof. Dr. Marcos Fagundes Caetano
CIC/UnB

Prof. Dr. Marcelo Grandi Mandelli
Coordenadora do Bacharelado em Ciência da Computação

Brasília, 24 de julho de 2023

Dedicatória

Dedico este trabalho à minha futura esposa Débora que esteve ao meu lado durante todo o período de realização deste trabalho, oferecendo suporte, compreensão e apoio durante várias madrugadas, sempre se preocupando com meu bem estar me levantando nos momentos mais difíceis.

E também aos meus queridos pais Júlio e Regina, que me deram a vida e pavimentaram todo o caminho para que eu pudesse chegar onde eu estou hoje.

Agradecimentos

Agradeço ao meu companheiro de projeto Klayton Castro e todos os servidores do STF que auxiliaram na realização deste projeto, pela confiança no meu trabalho e por todo conhecimento passado durante meus dois anos de estágio.

Agradeço também à minha psicóloga Cassiana, não só pelo apoio psicológico durante todo o período de realização deste trabalho, mas também por ter me apresentado ferramentas que foram extremamente úteis para a escrita deste trabalho.

Agradeço ao meu amigo Pedro, por ter me auxiliado nas edições e traduções das imagens deste trabalho e, por sempre se mostrar disponível à ajudar.

Por fim, agradeço especialmente à Prof.a Dr.a Aletéia Patrícia, por ter aceitado ser minha orientadora durante a realização deste trabalho, por toda ajuda e conhecimentos a mim passados, pela paciência e disponibilidade nos momentos difíceis e, por tornar este trabalho possível.

Resumo

O modelo de computação em nuvem trás vários benefícios à organizações e empresas que o utilizam, como o rápido provisionamento de recursos de infraestrutura e a capacidade de hospedar aplicações de maneira distribuída, com alta disponibilidade e confiabilidade. No entanto, observa-se que há uma grande dificuldade de se implantar um modelo de computação em nuvem, especialmente em organizações do setor governamental brasileiro, possivelmente devido à utilização de *softwares* e modelos de infraestrutura legados, sem padronização para a integração de sistemas. Como solução para auxiliar na transição entre uma arquitetura legada para a arquitetura de computação em nuvem, a primeira versão da plataforma de gerenciamento de nuvem Cloud.Jus foi desenvolvida entre os anos 2017 e 2019 para realizar o gerenciamento de recursos de infraestrutura do ambiente computacional do Supremo Tribunal Federal (STF), implantando efetivamente um modelo de computação em nuvem para uso dos funcionários do órgão governamental. No entanto, observou-se a oportunidade de realizar melhorias na plataforma Cloud.Jus, a fim de realizar o transporte e organização de dados de maneira mais eficiente, além de tornar cada componente da plataforma mais desacoplado e modular. Visto isso, são apresentados as ferramentas utilizadas e melhorias implementadas na plataforma Cloud.Jus versão 3.0, além dos testes realizados entre as duas versões da plataforma, mostrando um aumento de eficiência e uma grande redução no tempo total de execução de requisições após a implantação das melhorias.

Palavras-chave: Computação em Nuvem, Gerenciamento de Recursos, Plataforma de Gerenciamento de Nuvem

Abstract

The cloud computing model brings several benefits to organizations and companies that use it, such as easy provisioning of infrastructure resources and the ability to host applications with high availability and reliability in a distributed manner. However, there is great difficulty in deploying a cloud computing model, especially in organizations in the Brazilian government sector, possibly due to the use of legacy software and legacy infrastructure models with no standardization for systems integration. As a solution to assist in the transition from a legacy architecture to a cloud computing architecture, the first version of the Cloud.Jus cloud management platform was developed between 2017 and 2019 to manage the infrastructure resources of the computational environment of the Brazilian Federal Supreme Court, effectively deploying a cloud computing infrastructure to be used by government employees. However, there was an opportunity to improve the platform, making data transmission and organization inside the system more efficient. Also, the upgrades would make each platform component more decoupled and modular. This work presents the tools and improvements made in the Cloud.Jus platform and the tests carried out between the prototype and the new version of the platform to validate the increase in efficiency and a large reduction in the total execution time after implementing the upgrades.

Keywords: Cloud Computing, Resources Management, Cloud Management Platform

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

API *Application Programming Interface.*

AWS Amazon Web Services.

CMP *Cloud Management Plataform.*

DBaaS *Database as a Service.*

DMZ *Demilitarized Zone.*

DNS *Domain Name System.*

GCP Google Cloud Plataform.

HTTP *HyperText Transfer Protocol.*

HTTPS *Hypertext Transfer Protocol Secure.*

IaaS *Infrastructure as a Service.*

IP *Internet Protocol.*

MLaaS *Machine Learning as a Service.*

NAT *Network Address Translation.*

NIST *National Institute of Standards and Technology.*

PaaS *Plataform as a Service.*

phpIPAM *(IP Address Management.*

PJU Poder Judiciário da União.

SaaS *Software as a Service.*

SLA *Service Level Agreements*.

SSH *Secure Socket Shell*.

STF Supremo Tribunal Federal.

TI Tecnologia da Informção.

VM *Virtual Machine* - Máquina virtual.

Capítulo 1

Introdução

Atualmente, observa-se um crescimento da complexidade dos setores de Tecnologia da Informação (TI) de grande parte das organizações e empresas. Com o aumento da complexidade destes setores, há uma necessidade cada vez maior de equipes especializadas no gerenciamento e na administração dos setores de TI, especialmente, no quesito de atribuição de recursos de infraestrutura [20]. Assim sendo, percebe-se uma necessidade cada vez maior de desenvolver soluções que sejam capazes de integrar o ambiente tecnológico como um todo, especialmente, tratando-se de organizações compostas por diferentes times, cada um com seu ambiente de trabalho e ferramentas específicas que são utilizadas para a realização das suas atividades [21].

Assim sendo, o modelo de computação em nuvem surgiu, entre outros motivos, para integrar ambientes computacionais complexos e facilitar a gerência e o provisionamento de recursos computacionais. Atualmente, a utilização do paradigma de computação em nuvem se encontra em ascensão, não só em relação à industria de tecnologia, como também em todos os setores do mercado. [22, 23]. Computação em nuvem consiste na manipulação de recursos computacionais como serviço, de forma que, o provisionamento de tais recursos pode ser feito por usuários sob-demanda, com o mínimo de esforço de gerenciamento, seguindo acordos pré-determinados entre o provedor de serviços em nuvem e o cliente [24, 25].

No entanto, percebe-se uma certa dificuldade por parte das organizações em realizar a migração de um sistema de infraestrutura legada para o modelo de computação em nuvem [26]. Em especial, este trabalho apresenta como exemplo a situação dos órgãos do Poder Judiciário da União (PJU) que fazem utilização de softwares legados, além de possuirem um ambiente de infraestrutura consolidado e heterogêneo. Em contrapartida, é de interesse destas organizações a implantação de um ambiente em nuvem, para que seja possível que estas organizações possam usufruir dos benefícios que o modelo de computação em nuvem oferece.

A fim de sanar esse problema, foi proposta uma ferramenta para auxiliar na transição entre o modelo de infraestrutura legada e o modelo de computação em nuvem. A ferramenta proposta, chamada Cloud.Jus [27], foi desenvolvida e implementada para ser utilizada como uma plataforma de gerenciamento em nuvem pelos administradores do ambiente de infraestrutura e funcionários do setor de Tecnologia da Informação do Supremo Tribunal Federal (STF) [27]. A ferramenta proposta foi construída para integrar o ambiente de infraestrutura legado e heterogêneo do STF, facilitando o provisionamento de recursos de infraestrutura por parte dos administradores de infraestrutura e usuários da plataforma, implantando um modelo de computação em nuvem.

Entretanto, observou-se que certos componentes e métodos utilizados na primeira versão da plataforma Cloud.Jus, desenvolvida entre 2017 e 2019, poderiam ser melhorados e atualizados utilizando ferramentas, métodos e soluções mais modernas e sofisticadas. Assim, notou-se que era possível a implementação de novos componentes e a utilização de ferramentas modernas a fim de melhorar a comunicação entre componentes, armazenamento de dados, e tornar a plataforma mais modular e desacoplada. Essa modernização na plataforma Cloud.Jus facilita a sua exportação para outras organizações do Poder Judiciário da União, tornando possível a construção de uma nuvem comunitária do judiciário. Diante do exposto, este trabalho propõe uma modernização da plataforma de gerenciamento de nuvem Cloud.Jus, utilizando métodos e soluções do estado da arte.

1.1 Objetivos

O objetivo geral deste trabalho é desenvolver e implementar melhorias a fim de facilitar o transporte e organização de informações e dados trafegados na plataforma Cloud.Jus, utilizando ferramentas e métodos do estado da arte. Assim, para que o objetivo geral deste trabalho seja cumprido, faz-se necessário atingir os seguintes objetivos específicos:

- Desenvolver uma API (*Application Programming Interface*) para funcionar como mediador entre a interface de usuário e as camadas mais internas da plataforma;
- Implantar um banco de dados integrado com a API para armazenamento e organização de dados e informações necessárias para o bom funcionamento da plataforma;
- Implantar um serviço de mensageria distribuída para auxiliar no transporte e balanceamento de carga de requisições entre os componentes de *backend*.

1.2 Organização deste Documento

A fim de apresentar os conceitos necessários para a apresentação da plataforma Cloud.Jus, e em seguida as propostas de melhoria da ferramenta, este trabalho foi dividido em cinco capítulos. O Capítulo 2 introduz os conceitos de computação em nuvem, apresentando desde as suas características essenciais e modelos utilizados, até os principais provedores de serviços em nuvem atuantes no mercado atualmente. O Capítulo 3 apresenta a definição de plataforma de gerenciamento em nuvem, exemplificando algumas das plataformas utilizadas no mercado, e introduzindo os conceitos necessários para o entendimento da função e funcionamento da plataforma de gerenciamento Cloud.Jus. Na sequência, o Capítulo 4 apresenta as principais ferramentas e tecnologias em nuvem utilizadas na implementação da plataforma e nas melhorias adicionadas. No Capítulo 5 é apresentada a visão geral da arquitetura da plataforma Cloud.Jus, e também é feito um aprofundamento nas soluções e melhorias propostas para a nova versão da plataforma Cloud.Jus. Por fim, o Capítulo 6 apresenta as conclusões alcançadas neste trabalho e destaca alguns trabalhos futuros.

Capítulo 2

Computação em Nuvem

Neste capítulo serão abordadas as definições e as principais características da computação em nuvem. A Seção 2.1 apresentará a definição de computação em nuvem. Na sequência, a Seção 2.2 descreve as principais funcionalidades e as características essenciais da computação em nuvem. As Seções 2.3 e 2.4 listam os modelos de implantação e os modelos de serviço em nuvem, respectivamente. A Seção 2.5 aborda os principais provedores de serviços em nuvem, enquanto a Seção 2.6 discorre sobre a abordagem de nuvens federadas. Por fim, a Seção 2.7 apresenta as considerações finais deste capítulo.

2.1 Definição

Apesar de ser um assunto e uma abordagem relativamente recente no mercado, o conceito subjacente de *cloud computing* foi enunciado em 1961 por John McCarthy quando ele afirmou que "*chegará um tempo em que os sistemas serão fortemente orientados a serviços e a computação poderá ser definida como um serviço de utilidade pública*"[28]. Em 2011, o *National Institute of Standards and Technology* (NIST) - Instituto Nacional de Padrões e Tecnologia dos Estados Unidos, definiu computação em nuvem como "um modelo para habilitar processo onipresente e sob demanda à rede de um conjunto compartilhado de recursos de computação configuráveis (por exemplo, redes, servidores, armazenamento, aplicativos e serviços), que podem ser provisionados e liberados com o mínimo de esforço de gerenciamento ou iteração do provedor de serviços" [29].

Conforme os recursos computacionais foram se tornando cada vez mais acessíveis e baratos [30], mais a computação em nuvem foi se difundindo, facilitando a escalabilidade de recursos computacionais, alimentando a ilusão de recursos virtualmente infinitos. Desde então, a computação em nuvem vem se tornando uma alternativa flexível e com uma relação custo-benefício melhor em comparação à solução tradicional, na qual todo o trabalho computacional é feito localmente. A difusão da computação em nuvem per-

mite que as organizações escalem seus recursos computacionais de forma fácil e rápida, pagando apenas pela quantidade de recurso sendo consumido, e reduzindo a necessidade e a dependência dos recursos de hardware e infraestrutura local [31].

Com o início da pandemia da Covid-19, quase todos os setores da sociedade, desde pequenas e grandes companhias, até os setores de saúde e educação, tiveram que se adaptar e converter diversos de seus serviços para o ambiente digital [32]. Com isso, a utilização e o volume de dados gerados por aplicações em nuvem cresceu fortemente, aumentando os desafios nas áreas de segurança, utilização de dados privados e estabilidade para serviços e aplicações em nuvem.

Visto isso, com o advento desta nova tecnologia, percebeu-se uma necessidade cada vez maior de definir conceitos formais e características essenciais que definem o que é um sistema computacional em nuvem [33]. Desta forma, instituições como o NIST, definiram os principais paradigmas e características que um sistema deve conter para ser considerado como um sistema computacional em nuvem [25].

2.2 Características Essenciais de Nuvem

Em 2011, o NIST [25] definiu certos paradigmas e características essenciais para uma plataforma computacional ser considerada como um sistema em nuvem. Tais paradigmas podem ser identificados por apresentarem cinco características essenciais, três modelos de serviços e quatro tipos de implementação. Assim, segundo o NIST [25] as principais características de nuvem são:

- **Serviços sob Demanda:** recursos computacionais são disponibilizados de acordo com a demanda do cliente e disponibilidade do provedor, de tal forma que tais recursos são consumidos uniliteralmente;
- **Facilidade de Acesso:** os recursos devem estar disponíveis de forma fácil para o cliente, podendo ser acessados por um amplo espectro de dispositivos, desde que tenham acesso à rede mundial de computadores (Internet). Os clientes podem consumir processamento, armazenamento e rede de forma independente, sem necessidade de interação humana com o provedor de serviços;
- **Elasticidade:** a capacidade de nuvem deve ser facilmente e rapidamente alterada, e a alocação e a desalocação de recursos devem ser feitas em tempo de execução, sem prejuízo ao funcionamento do sistema de acordo com a demanda do cliente;
- **Medição de Serviço:** a plataforma em nuvem deve conter ferramentas de monitoramento do uso de recursos. Além disso, os dados de monitoração devem ser

transparentes, tanto para o cliente quanto para o provedor, a fim de ser possível monitorar, auditar e relatar tais dados.

2.3 Modelos de Serviço

Assim como os modelos de implantação, NIST [25] também definiu os modelos de serviço que especificam o nível de abstração das soluções baseadas em computação em nuvem. Tais modelos são definidos como IaaS, PaaS, SaaS, descritos a seguir:

- **IaaS** - *Infrastructure as a Service*: Infraestrutura como Serviço é o nome dado para os serviços oferecidos ao consumidor em forma de recursos, tais como servidores, rede, armazenamento ou outros recursos computacionais. Esse tipo de serviço objetiva opta a construção de um ambiente de aplicação sob demanda, capaz de suportar aplicativos ou sistemas operacionais. De forma geral, o cliente não tem controle direto sobre a estrutura de nuvem, no entanto, o mesmo tem controle total sobre os aplicativos ou sistemas operacionais funcionando sob tal infraestrutura;
- **PaaS** - *Plataform as a Service*: Plataforma como Serviço é o nome dado a um tipo de serviço de alto nível, no qual o usuário não tem controle sobre a infraestrutura ou mesmo sobre os componentes da mesma, incluindo rede, servidores ou sistemas operacionais. A função de uma plataforma como serviço é disponibilizar ao usuário serviços que viabilizem a implementação, e o desenvolvimento de aplicações na nuvem, sem preocupação com as características de mais baixo nível da infraestrutura. Os serviços tipo PaaS oferecem ao cliente um sistema operacional, linguagens de programação e ambientes de desenvolvimento de aplicações, contendo ferramentas para auxiliar no desenvolvimento e implementação de aplicações;
- **SaaS** - *Software as a Service*: Software como Serviço é o nome dado a softwares prontos e já em funcionamento em nuvem para um usuário final. Tais aplicações, geralmente, são acessíveis por meio de interfaces como APIs – *Application Programming Interface* – ou navegadores web. O cliente não administra e nem tem controle sobre a infraestrutura ou componentes de rede, servidores, armazenamento ou sistemas operacionais, e nem das características da aplicação, exceto algumas configurações específicas.

Com os avanços e o amadurecimento das tecnologias em nuvem, novos serviços têm sido ofertados pelos provedores, causando uma necessidade de redefinição das fronteiras dos modelos de serviços definidas pelo NIST [25]. Modelos de serviço, como o SaaS, vêm sendo renomeados e divididos em várias categorias, a fim de representar de forma mais clara a função de cada tipo de serviço.

Para representar melhor a diversificação dos serviços oferecidos, os provedores de soluções em nuvem começaram a utilizar o termo "XaaS" (*Everything as a Service*) [34]. No entanto, por se tratar de uma nomenclatura relativamente nova, não existe um padrão para o nome dos modelos utilizados pelos diversos provedores de soluções em nuvem, o que acaba causando uma certa confusão. Assim, pode ocorrer de um modelo ofertado por um provedor ter a mesma sigla adotada por outro provedor, mas entregar soluções diferentes. Assim, são listados a seguir exemplos de modelos de serviço oferecidos atualmente, incluindo os modelos com nomenclatura igual, mas que se tratam de soluções diferentes [35, 36].

- **AaaS:** *Analytics as a Service*, ferramentas analíticas como serviço [37];
- **BaaS:** *Backup as a Service*, cópia de segurança como serviço [38];
- **BaaS:** *Banking as a Service*, serviços bancários como serviço em nuvem [39];
- **CaaS:** *Container as a Service*, contêiner como serviço [40];
- **DaaS:** *Desktop as a Service*, área de trabalho como serviço [36];
- **DaaS:** *Data as a Service*, dados como serviço [35];
- **DBaaS:** *Database as a Service*, anco de dados como serviço [41];
- **FaaS:** *Function as a Service*, função como serviço [42];
- **MaaS:** *Monitoring as a Service*, monitoração como serviço [37];
- **MLaaS:** *Machine Learning as a Service*, aprendizado de máquina como serviço [43, 44];
- **STaaS:** *Storage as a Service*, armazenamento como serviço [45, 46];
- **SecLaas:** *Secure Logging as a Service*, registro seguro como serviço [47].

Por se tratarem de nomenclaturas e tipos de serviços novos, não há um padrão para a forma de entrega e a qualidade do serviço oferecido pelos diversos provedores. Assim sendo, fica a cargo do cliente conhecer detalhadamente o que cada provedor oferece, a fim de evitar problemas futuros.

2.4 Modelos de Implantação

O NIST [25] também definiu quatro formas de implantação de uma plataforma em nuvem, os quais são nuvens públicas, nuvens privadas, nuvens comunitárias e nuvens híbridas. Tais tipos de implantação são apresentadas a seguir:

- **Nuvem Pública:** os recursos podem ser acessados pelo público em geral, e devem haver ferramentas de controle de acesso a fim de atender as demandas e manter a segurança dos dados dos clientes. A plataforma pode ser de propriedade de uma organização comercial, acadêmica ou governamental, ou alguma gerência híbrida delas [24];
- **Nuvem Privada:** a nuvem privada é utilizada exclusivamente por uma organização, mas ela pode ser gerada ou hospedada por terceiros, ou pela própria instituição. O ambiente deve ser hospedado dentro das instalações da empresa (*on-premises*);
- **Nuvem Comunitária:** a infraestrutura é compartilhada por diversas organizações, as quais compartilham certos requisitos, tais como: missão, aspectos de segurança, politica e conformidade. A plataforma em nuvem pode ser implantada de maneira local ou remota, e seu gerenciamento e operação pode ser feito por uma ou mais organizações, terceiros ou uma combinação destes;
- **Nuvem Híbrida:** uma arquitetura de nuvem híbrida é definida como uma combinação de duas ou mais arquiteturas de nuvem distintas (privadas, comunitárias ou públicas), na qual permanecem como entidades exclusivas, mas integradas por tecnologias proprietárias ou padronizadas, viabilizando a portabilidade de dados e de aplicações entre os ambientes.

Atualmente, com o crescimento de empresas como a AWS, Microsoft e Google no mercado de computação em nuvem, a popularidade de nuvens públicas vem crescendo mais a cada dia [48]. Obeserva-se um interesse crescente no mercado de computação em nuvem e cada vez mais empresas do ramo de tecnologia vêm se interessando no mercado, oferecendo serviços e recursos virtuais a instituições e clientes interessados [49]. Na próxima seção serão apresentados os principais provedores de nuvem e serviços oferecidos atualmente no mercado de computação em nuvem.

2.5 Provedores de Serviços

Provedores públicos, são provedores de serviços em nuvem que alugam seus serviços e recursos para o público em geral. Existem diversos provedores publicos atuando atualmente no mercado, disponibilizando armazenamento e processamento a custos acessíveis para diversos usuários. Segundo Bittman [50], uma vez que cada usuário paga pelo que consome dos recursos e serviços, é possível dividir o custo de manutenção, suporte e operação para os diversos clientes que usufruem da infraestrutura em nuvem.

Assim sendo, são comumente utilizadas *Service Level Agreements* (SLAs) – Acordos de Nível de Serviço [51] para determinar a relação do provedor de serviço e o cliente. No

SLA são redigidos os requisitos de entrega de serviços, tal como percentual de garantia de disponibilidade. Cada provedor utiliza um SLA com termos próprios, e caso não haja o cumprimento dos termos redigidos, o cliente pode ter direito a ressarcimento de custos, ou em outros casos acarretar multas ao provedor, dependendo dos termos do contrato.

Figura 2.1: Quadrante Mágico de Plataforma em Nuvem 2022 [1].

Todos os anos o Instituto Gartner [52] realiza estudos de diferentes mercados a fim de comparar a atuação das principais empresas, de acordo com sua área de atuação. O chamado "Quadrante Mágico" é um relatório anual que é a culminação dos estudos em mercado específico, e as principais empresas que atuam em tal mercado. O relatório Quadrante Mágico do Gartner é dividido em 4 quadrantes (conforme apresentado na Figura 2.1), sendo eles:

- **Líderes:** competidores que executam bem sua visão de mercado e estão bem posicionados para o futuro;
- **Visionários:** competidores que têm um entendimento ou uma visão de futuro do mercado onde atuam, mas ainda têm desafios para executar bem neste mercado;

- **Concorrentes de Nicho:** competidores que atuam bem em um segmento pequeno de mercado, ou não têm um foco específico e não inovam ou performam tão bem quanto outros competidores;
- **Desafiantes:** competidores que têm uma boa execução ou até mesmo dominam um segmento grande do mercado, mas não têm uma visão de futuro do mercado.

Dessa forma, é possível observar na Figura 2.1 que em relação aos anos anteriores, os três líderes de mercado: AWS, Microsoft Azure e GCS, permanecem com o domínio maior de mercado, enquanto antigos competidores de nicho como Alibaba Cloud e Oracle continuam seu crescimento no mercado de computação em nuvem, atingindo a classificação de visionários [1]. É possível observar também que novos competidores de nicho vêm ganhando espaço no mercado como o *Tencent Cloud* e *Huawei Cloud*.

2.5.1 Amazon Web Services (AWS)

Amazon Web Service (AWS) [53], fundada em 2006, é o setor tecnológico da empresa Amazon. Por ser a primeira empresa a entrar no mercado de computação em nuvem, consumidores tendem a escolher a AWS não só pelo seu tempo de mercado, mas também pela confiabilidade e estabilidade dos produtos e serviços [54]. Atualmente, a AWS é o maior provedor de soluções e produtos em nuvem do mundo, superando outros líderes do mercado de computação em nuvem [55].

Estando presente em 99 zonas de disponibilidade, espalhadas por 31 regiões geográficas por todo mundo [2], como mostra a Figura 2.2, a AWS oferece soluções para diversas áreas, tais como publicidade e *marketing*, serviços financeiros, tecnologia de jogos e órgãos governamentais [56]. A AWS também oferece soluções em nuvem para as áreas de análises de *datalakes, machine learning*, computação em nuvem sem servidor e armazenamento [57].

A AWS também oferece diversos serviços e produtos de computação em nuvem que estão disponíveis para o público em geral, os principais produtos e serviços são:

- **Amazon Lightsail:** serviço de PaaS que oferece servidores, armazenamento, banco de dados e arquitetura de rede totalmente virtuais a fim do cliente desenvolver, implementar e executar aplicações em nuvem [58];
- **Amazon EC2:** serviço de IaaS, ofertando máquinas virtuais e toda uma estrutura de infraestrutura em nuvem [59];
- **Amazon S3:** o Amazon Simple Storage é um serviço de armazenamento de dados do tipo armazenamento por objeto, em nuvem [60];

Figura 2.2: Mapa da infraestrutura Global da AWS [2].

- **Amazon Aurora:** serviço de gestão de banco de dados, oferecendo um sistema de gerenciamento de banco de dados relacional [61];
- **Amazon DynamoDB:** serviço de DBaaS, oferecendo serviço de banco de dados não relacional em nuvem [62].

2.5.2 Microsoft Azure

Microsoft Azure, lançada em 2010, é a principal plataforma de nuvem oferecida pela empresa Microsoft [63]. Tendo como suas maiores concorrentes a AWS e a GCP, a maior vantagem que o Azure tem sobre seus concorrentes é oferecer os produtos da Microsoft em nuvem para uso do público em geral, como por exemplo os produtos da OpenAI [64].

A infraestrutura da Microsoft Azure está disponível por mais de 60 regiões ao redor do mundo, sendo assim a plataforma disponível em mais regiões do mundo [65]. Na Figura 2.3 é possível observar, em azul, as regiões onde o Microsoft Azure está disponível, enquanto os marcadores em cinza representam *datacenters* que estarão disponíveis para oferecer serviços em breve [3].

Atualmente, o Azure disponibiliza mais de 200 produtos e serviços [63], dentre eles, os principais oferecidos são:

Figura 2.3: Mapa da infraestrutura global da Microsoft Azure [3].

- **Azure OpenAI Service:** serviço de geração de modelos de inteligência artificial oferecidos pela OpenAI [66];
- **Azure SQL:** serviço DBaaS que oferece bancos relacionais em nuvem [67];
- **Azure Cognitive Service for Vision:** serviço de análise de imagem e pesquisa visual computacional, utilizando inteligência artificial e OCR [68];
- **Azure Machine Learning:** serviço de MLaaS (*Machine Learning as a Service*), aprendizado de máquina como serviço para utilização corporativa [69];
- **Container Instances:** serviço de PaaS que oferece soluções em *container* para desenvolvimento e execução de aplicativos [70];
- **Microsoft Defender for Cloud:** soluções de segurança e proteção de dados para serviços em nuvem [71].

2.5.3 Google Cloud Plataform (GCP)

Lançada em 2008, a Google Cloud Plataform (GCP) é a plataforma de nuvem da Google [72]. Com o objetivo de competir diretamente com outros concorrentes como AWS e Azure, a GCP oferece diversos serviços em nuvem para seus clientes. Uma das ferramentas em nuvem oferecidas pela GCP é o popular Google Docs [73], uma alternativa gratuita ao Microsoft Office, da Microsoft, concorrente direto da Google no mercado de computação em nuvem.

Desde 2008, a Google vem utilizando sua poderosa infraestrutura para fornecer serviços e produtos para o público, a fim de consolidar sua dominância no mercado de computação em nuvem. Atualmente, a GCP possui 112 zonas de disponibilidade, abrangendo 37

regiões, atuando em mais de 200 países e territórios, como apresentado na Figura 2.4 [4]. A GCP também oferece mais de 150 produtos, dentre eles, diversos produtos gratuitos para diferentes áreas, tais como infraestrutura em nuvem, armazenamento em nuvem, inteligência artificial, aprendizado de máquina, banco de dados em nuvem, análise de dados, infraestrutura de rede e ferramentas para desenvolvedores.

Figura 2.4: Mapa da infraestrutura global da GCP [4].

2.5.4 Outros Provedores

Apesar da grande dominância de mercado dos provedores listados nas Seções 2.5.1, 2.5.2 e 2.5.3, estes estão longe de serem os únicos competidores que atuam no mercado de computação em nuvem. Muitos provedores vêm ganhando cada vez mais espaço no mercado, tornando-se competidores visionários como mostra o estudo do Quadrante Mágico de Gartner de 2022, enquanto surgem mais competidores de nicho disputando espaço no mercado [1]. Dois exemplos que devem ser mencionados são os provedores Alibaba Cloud e Oracle, que vêm tendo uma atuação e crescimento cada vez maior no mercado, Logo, em poucos anos esses competidores devem estar na categoria de líderes de mercado, assim como outros provedores que vêm disputando espaço no mercado:

- **Alibaba Cloud:** fundada em 2011, Alibaba Cloud, também chamada de Aliyum, é um provedor de soluções de nuvem fundado na China [74]. Apesar de fornecer

serviços e soluções em nuvem para o mercado majoritariamente chinês, Alibaba Cloud vêm tendo um crescimento de mercado bem expressivo, como mostra os relatórios do Quadrante Mágico de Gartner dos últimos anos [1]. Atualmente, o Alababa Cloud conta com 86 zonas de disponibilidade atuando em 28 regiões [74];

- **Oracle Cloud:** disponibilizada em 2016 pela empresa Oracle [75], assim como o Alibaba Cloud, vêm tendo um crescimento significativo no mercado de computação em nuvem, passando de um competidor de nicho em 2021 para um dos competidores visionários segundo o relatório do Quadrante Mágico de Gartner [1]. Atualmente, a Oracle Cloud abrange 41 regiões, e oferece mais de 100 soluções e serviços de plataforma e infraestrutura [75];
- **Tencent Cloud:** situado na China, Tencent Cloud é o braço da computação em nuvem do conglomerado Chines Tencent [76]. Atualmente, a Tencent Cloud conta com 70 zonas de disponibilidade, atuando em 26 regiões;
- **IBM Cloud:** fundada em 2013 pela IBM, uma empresa já consolidada no mercado de computadores, a IBM Cloud vêm atuando desde então como competidor de nicho, focando em soluções de inteligência artificial e aprendizagem de máquina [77]. No entanto, a IBM Cloud também oferece mais de 170 produtos e serviços para infraestrutura em nuvem, plataforma em nuvem, entre outras áreas;
- **Huawei Cloud:** fundada pela empresa chinesa Huawei [78], a Huawei Cloud é uma das mais novas competidoras de nicho do mercado de computação em nuvem, como mostra o relatório do Quadrante Mágico de Gartner de 2022 [1].

2.6 Federação em Nuvens

Uma das ideias defendida por diversos pesquisadores é que para que os serviços em nuvem tenham um bom funcionamento é necessária uma infraestrutura flexível, interconectada e sempre presente, como são, por exemplo, os serviços de eletricidade ou telefonia [79]. Pesquisadores da área têm focado na integração entre nuvens, a fim de obter aprimoramentos na qualidade, confiabilidade e tornar as integrações menos custosas. Observa-se assim um crescimento na importância da interoperabilidade e portabilidade entre nuvens, de forma que usuários possam utilizar diversas nuvens integradas de maneira segura e com uma maior utilidade [5].

A federação de nuvens computacionais, também chamada de *inter-cloud* ou *cross-cloud*, é definida como a área de pesquisa dentro da computação em nuvem que estuda a comunicação de diversos provedores de nuvens computacionais, públicas, privadas ou

comunitárias, conectadas por meio da Internet [80]. Celesti *et al.* explicam [81], que a evolução do mercado da computação em nuvem pode ser dividida em três fases:

- **Fase Monolítica:** os serviços de computação em nuvem são baseados em arquiteturas proprietárias ofertadas por grandes provedores;
- **Fase da Cadeia Vertical de Fornecimento:** certos provedores utilizam serviços em nuvem de outros provedores, como fabricantes de software movendo aplicações para o modelo SaaS em nuvens públicas. Ainda existem ambientes proprietários, mas já existem iniciativas de federação entre nuvens;
- **Fase da Federação Horizontal:** provedores de porte médio ou pequeno colaboram entre si para obter um nível de escalabilidade e eficiência maior no uso de seus recursos.

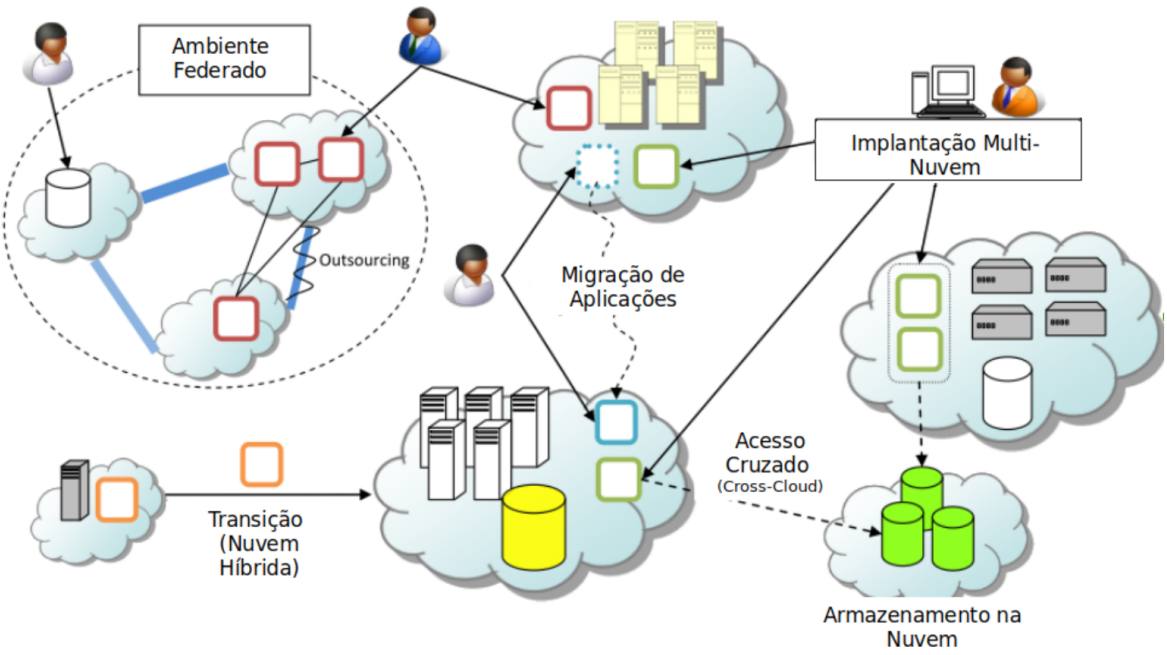

Figura 2.5: Relações de Interoperabilidade entre nuvens [5].

Como mostrado na Figura 2.5, nuvens federadas têm capacidade de realizar a migração cruzada de recursos, permitindo que seja implementada redundância e complementação de funcionalidades entre ambientes de nuvem de forma colaborativa. A vantagem de federar diversos ambientes em nuvem é obter uma maior amplitude e combinação de poder computacional por meio do compartilhamento de recursos em diferentes camadas [5].

Grozev e Buyya [79] afirmam que nuvens federadas, apesar de serem arquiteturas de nuvem descentralizadas por natureza, podem ter seus mecanismos gerenciados de forma

centralizada por um único provedor, de forma que todos os aspectos de integração são controlados e gerenciados por apenas um dos provedores que constituem a nuvem federada. Os mecanismos da nuvem federada podem também ser gerenciadas de forma *peer-to-peer* no qual existe a colaboração direta entre colaboradores da nuvem, tal que as particularidades de cada colaboração são definidas caso a caso.

Diversas abordagens podem ser utilizadas para realizar a federação entre dois ou mais provedores de serviços de nuvem. Segundo Toosi *et. al.* [5], para que exista a comunicação entre dois ambientes de nuvem podem ser utilizadas ferramentas para realizar a intermediação entre provedores, como *brokers* ou *middlewares*; implementar um padrão entre as interfaces dos ambientes em nuvem para que a comunicação seja possível; ou até mesmo um híbrido da padronização de interfaces e ferramentas de intermediação.

Em termos de interoperabilidade, cada cenário de federação de nuvens pode ser classificado de acordo com a função e o objetivo da federação. Os cenários podem ser categorizados como focados no cliente ou focados no provedor [5]. Nas abordagens com o foco no provedor, os mecanismos de interoperabilidade das nuvens federadas e nuvens híbridas são mantidos sob gerência dos próprios provedores envolvidos. Nas abordagens centradas no cliente, é da vontade do cliente gerenciar recursos de diversos provedores simultaneamente, para isso, são utilizadas ferramentas *multi-cloud*, capazes de fazer o gerenciamento de diversos provedores e fornecedores de serviço simultaneamente, funcionando como um *broker*, ou seja, uma que faz a intermediação entre o cliente e os provedores [5].

Atualmente, federações em nuvem vêm sendo amplamente implementadas utilizando nuvens públicas [82], no entanto é possível realizar a construção de uma federação de nuvem utilizando nuvens privadas. Diante disto, esse trabalho objetiva realizar melhorias e reestruturações no ambiente de gestão de nuvem privada do Supremo Tribunal Federal, que facilitará a construção de um ambiente federado entre provedores de nuvem de tribunais de justiça.

2.7 Considerações Finais

Neste capítulo foram apresentadas as definições de computação em nuvem assim como os principais provedores de serviços de nuvem pública. Além disso, foram também apresentados os tipos de implementação de federação de nuvem. Isso foi necessário a fim de introduzir os principais conceitos de computação em nuvem que fundamentais para o entendimento deste trabalho.

Assim, no próximo capítulo serão apresentadas as principais plataformas de gerenciamento de nuvem disponíveis hoje no mercado, assim como suas arquiteturas e métodos de implantação.

Capítulo 3

Plataformas de Gerenciamento de Nuvem

Existem cenários em que o cliente que consome serviços de provedores de soluções de nuvem necessita de ferramentas eficazes para auxiliá-lo na gerência do ambiente. Pode-se tomar como exemplo, clientes que utilizam serviços de diversos provedores simultaneamente a fim de compor o seu ambiente em nuvem. A fim de auxiliar na gerência de ambientes em nuvem, em especial, ambientes de nuvem hibrida, encontram-se disponíveis diversas opções de Plataformas de Gerenciamento de Nuvem (CMPs – *Cloud Management Plataforms*), as quais oferecem soluções para gerenciamento de nuvens privadas, publicas, comunitárias ou híbridas. Assim, esse capítulo tem como objetivo apresentar a definição de CMP, casos de uso e as principais CMPs sendo utilizadas hoje no mercado. A Seção 3.1 descreverá a definição de CMP enquanto na Seção 3.2 serão apresentados os principais CMPs disponíveis hoje no mercado. A seguir, na Seção 3.3 serão descritos os principais desafios na implantação de nuvens comunitárias no modelo IaaS (*Internet as a Service*) e na Seção 3.4 será apresentado uma revisão de literatura sobre diversos casos de uso de CMPs. Por fim, a Seção 3.5 conterá as considerações finais deste capítulo.

3.1 Definição

A popularização dos serviços em nuvem criou uma necessidade maior de recursos computacionais em nuvem e um maior suporte a *datacenters* elásticos em larga escala. Desta forma, a criação da arquitetura baseada em serviços proporcionou grandes ganhos de eficiência, facilitando a comunicação entre as camadas de abstração [20]. Tal avanço simplificou ainda mais a utilização do sistema por parte do usuário, visto que diversos aspectos da configuração, como localização física da infraestrutura em nuvem, passaram a ser de responsabilidade do provedor [83, 84].

Assim sendo, observou-se um crescimento no interesse em desenvolver Plataformas de Gerenciamento de Nuvem (CMPs – *Cloud Management Plataforms*) a fim de auxiliar o usuário a gerenciar a sua infraestrutura em nuvem, que pode ser composta por um ou mais provedores públicos, privados, ou até mesmo infraestrutura local (*on-premises*). Em especial, pode-se destacar as CMPs baseadas em código livre, que vêm se popularizando no mercado, tomando como exemplo a plataforma OpenStack [85] e OpenNebula [86], sendo estas, duas das principais plataformas de gerenciamento de nuvem de código aberto atualmente. O OpenStack se destaca, principalmente, por proporcionar uma ampla cobertura aos recursos das camadas de infraestrutura, utilizando uma arquitetura modular e utilizando projetos interoperáveis. O OpenNebula se destaca pela sua capacidade de gerenciar uma variedade de tipos de infraestrutura em nuvem, como nuvens privadas, públicas ou híbridas [85, 87].

Não obstante, outras plataformas além do OpenStack e OpenNebula vêm ganhando espaço no mercado de plataformas de gerenciamento de nuvem [88]. Plataformas como CloudStack [9] e Eucalyptus [89], mostram que a estruturação das soluções em nuvem de maneira econômica e eficiente é possível, mostrando-se alternativas viáveis aos provedores comerciais [90, 91, 92].

3.2 Principais CMPs

Com o objetivo de fazer a escolha correta de uma CMP, é necessário verificar a necessidade real do sistema e da nuvem implementada, avaliando e comparando os recursos ofertados por cada uma das opções de CMPs disponíveis, e a compatibilidade com as diversas plataformas e infraestruturas em nuvem. Visto isso, nesta seção serão listadas as principais plataformas de gerenciamento de nuvem, destacando seus recursos, funcionalidades, arquiteturas e compatibilidade com as principais arquiteturas em nuvem.

3.2.1 OpenStack

O OpenStack [93, 94, 6] é uma plataforma de gerenciamento de nuvem constituído por uma combinação de diversos softwares originalmente desenvolvidos pela NASA [95] e CSP Rackspace [96] e, atualmente, mantidos e continuamente evoluídos pela comunidade de código aberto. Com uma arquitetura modular e facilmente configurável, conforme mostrado na a Figura 3.1, a plataforma OpenStack oferece robustez e confiabilidade para gerenciar sistemas de nuvem públicas e privadas no modelo IaaS. Construída para se adequar aos diversos tipos hardware e recursos, o OpenStack pode ser utilizado como plataforma de gerenciamento de nuvem de pequeno a grande porte, adaptando-se ao tipo de ambiente e infraestrutura disponível.

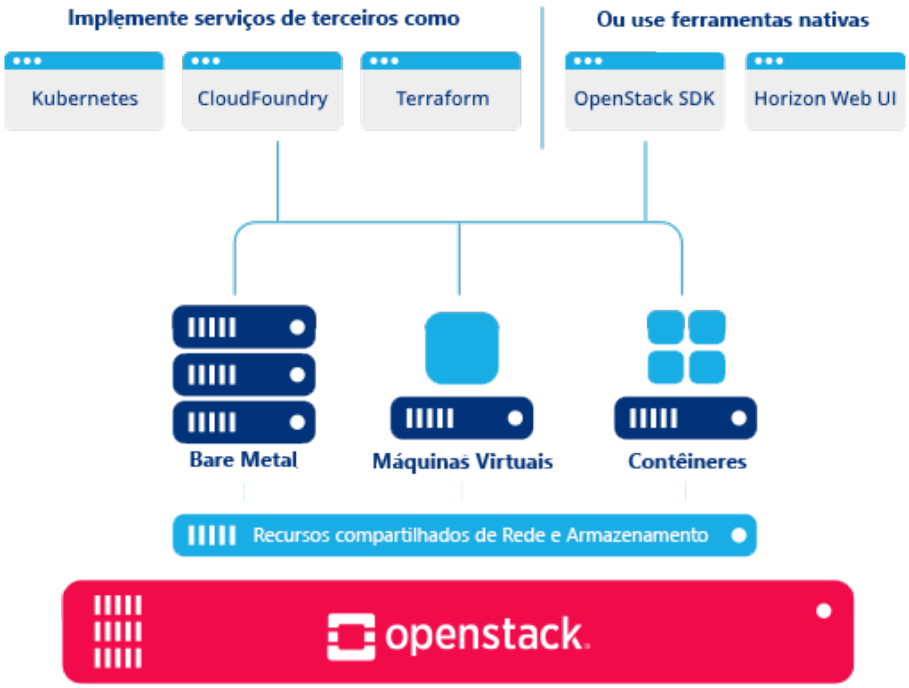

Figura 3.1: Arquitetura da Infraestrutura de Nuvem do OpenStack [6].

Além disso, a plataforma OpenStack contém diversos componentes de nuvem organizados em uma arquitetura orientada a serviços que suporta VMs, Contêineres, servidores físicos e toda a sua infraestrutura de armazenamento. O OpenStack utiliza APIs como ferramenta de comunicação, mantidos em uma rede compartilhada, de forma que os recursos e aplicações podem ser provisionadas remotamente, sem dependência de localidade [93]. Ademais, o OpenStack também contém um sistema de monitoramento, informando dados e métricas por meio de *Dashboards* de Gerenciamento e Monitoração [84].

Assim sendo, tem-se que a grande variedade de componentes, recursos e funcionalidades do OpenStack a torna uma plataforma maleável e adaptável a diversos tipos de sistemas e infraestruturas em nuvem, sendo capaz de atender diversas necessidades computacionais. A Figura 3.2, apresenta os principais componentes da plataforma OpenStack são [94, 6], os quais são:

- **Computação:**
  - **Nova:** interage com diversos hipervisores como VMware [97], Hyper-V [98], KVM [99] e Xen [7], fornecendo servidores virtuais. Embora haja certas limitações na integração com alguns hipervisores, como a necessidade da instalação de *plugins* proprietários, Nova oferece serviços de gerenciamento de recursos capazes de orquestrar instâncias, redes e controle de acesso por meio de APIs;

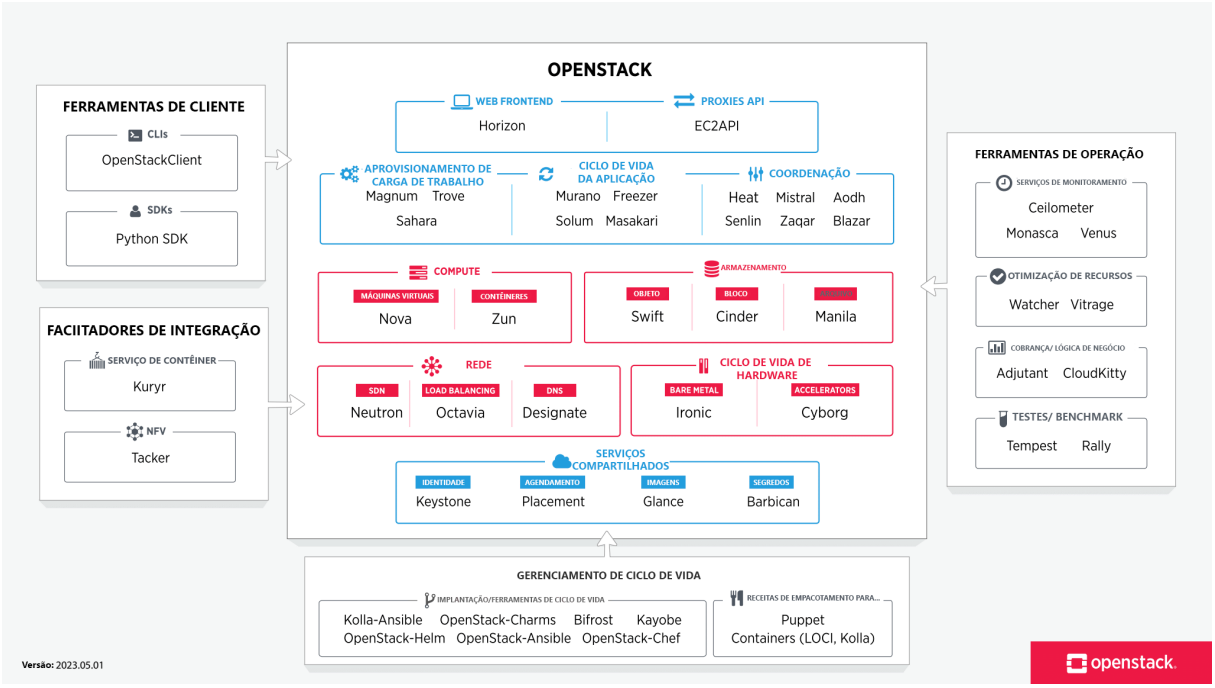

Figura 3.2: Arquitetura dos componentes principais do OpenStack [6].

  - **Zun:** provê uma API para gerenciar e inicializar contêineres baseados em diferentes tecnologias. Zun é útil para usuários que pretendem tratar conteineres como recurso gerenciado pelo próprio OpenStack, a fim de facilitar a integração com outros recursos nativos do OpenStack.

- **Ciclo de vida de Hardware:**
  - **Ironic:** implementa serviços e conjuntos de bibliotecas a fim de prover serviços escaláveis sob demanda, incluindo *bare metal*, VMs e contêineres;
  - **Cyborg:** é um acelerador de gerenciamento de ciclo de vida, oferecendo um *framework* de gerecimento de aceleradores como GPUs, FPGAs, etc.

- **Armazenamento:**
  - **Swift:** serviço de armazenamento de objetos ou dados tipo *blob*. Pode ser utilizado por organizações a fim de armazenar grandes quantidades de dados de maneira segura e barata. Ele foi construído a fim de ser altamente escalável e otimizado para durabilidade, disponibilidade e acesso concorrente, visto isso, Swift é extremamente útil para armazenamento de dados não-estruturados;
  - **Cinder:** serviço de armazenamento de dados em bloco. Cinder virtualiza a gerência de armazenamento de dados em bloco, oferecendo APIs para consumir e requisitar recursos de armazenamento, sem que o usuário necessite de

conhecimento sobre o local e o tipo de unidade de armazenamento que está sendo utilizada;

  - **Manila:** utilizado para prover acesso coordenado a sistemas compartilhados ou distribuídos.

- **Rede:**

  - **Neutron:** utiliza protocolos dinâmicos (DHCP), estáticos (IPs) ou redes virtuais (VLANs) a fim de fornecer conectividade entre dispositivos sob a gerência do OpenStack. A sua arquitetura permite que seus usuários usufruem dos componentes e ferramentas de segurança, como *firewalls*, sistemas de detecção e prevenção de invasão, balanceadores de carga, entre outros, já disponíveis na infraestrutura dos *datacenters*. Além disso, ele também oferece serviços de virtualização de rede;
  - **Octavia:** serviço de balanceamento de carga de código livre. Baseado no projeto neutron LBaaS (Balanceador de Carga como Serviço) e lançado na versão "Liberty" do OpenStack, Octavia foi projetada para fazer o balanceamento de carga de grandes quantidades de máquinas virtuais, contêineres, ou servidores *bare metal*;
  - **Designate:** provê serviços de DNS para a plataforma OpenStack.

- **Serviços Compartilhados:**

  - **Keystone:** funciona como o serviço de identidades do OpenStack, oferecendo serviços de autenticação de clientes, através de APIs. Keystone fornece ao usuário um sistema de *logon* único, implementado utilizando *tokens* de acesso. Além disso, o Keystone também oferece suporte para LDAP, Oauth, OpenID Connect, SAML e SQL;
  - **Placement:** serviço que oferece APIs baseadas no protocolo HTTP para monitorar o inventário de recursos disponíveis em uso, a fim de auxiliar os demais serviços a gerenciar e alocar recursos necessários para o seu funcionamento;
  - **Glance:** serviço de gerenciamento de imagens e *templates* de máquinas virtuais. Utiliza APIs para consultar os metadados das imagens das VMs, além do catálogo e gerenciamento das bibliotecas de imagens de servidor. As imagens guardadas pelo Glance podem ser armazenadas em diversas localidades, desde sistemas de ficheiro tradicionais até em sistemas de armazenamento de objetos, como o projeto do Swift, do próprio OpenStack;

  - **Barbican:** é o serviço de gerenciamento de chaves do OpenStack, oferecendo armazenamento seguro, provisionamento e gerência de dados secretos, como senhas, chaves de criptografia, Certificados X.509 e até mesmo dados binários crus.

- **Orquestração:**

  - **Heat:** é o orquestrador base do OpenStack. Heat realiza a orquestração dos recursos de infraestrutura para aplicações baseadas em *templates* em formato de texto que podem ser tratados como código. Heat utiliza APIs ReST e APIs Query para seu gerenciamento, além de oferecer serviços de escalonamento automáticos capazes de se comunicar com os serviços de telemetria, caso o usuário queira tratar um grupo de escalonamento como recurso em um *template*;
  - **Senlin:** é o serviço de *clustering* do OpenStack. A sua função é criar e operar *clusters* de objetos homogêneos expostos aos demais serviços do OpenStack, a fim de facilitar a orquestração de coleções de objetos similares;
  - **Mistral:** é o serviço de carga de trabalho do OpenStack, com a função de gerenciar a ordem de trabalho dos processos e relações de trabalho entre processos utilizando arquivos escritos em linguagem baseada em YAML, a fim do serviço tomar conta do gerenciamento de estado, execução na ordem correta, paralelismo, sincronização e alta disponibilidade;
  - **Zaqar:** é o serviço de mensageria do OpenStack desenvolvido para auxiliar desenvolvedores web e *mobile*. Zaqar utiliza APIs REST a fim de permitir que desenvolvedores enviem mensagens entre os diversos componentes das suas aplicações *mobile* e seus componentes SaaS. Além disso, Zaqar é composto por um motor de mensageria robusta e eficiente, projetada com alta escalabilidade e segurança, além da capacidade de ser integrada com outros componentes do OpenStack;
  - **Blazar:** é um serviço de reserva de recursos que permite usuários reservar quantidades ou tipo de recursos específicos por uma determinada quantidade de tempo;
  - **AODH:** é um serviço de alarme do OpenStack que executa certas ações baseadas em gatilhos específicos definidos pelos usuários da plataforma.

- **Provisionamento de Carga de Trabalho:**

  - **Magnum:** é o motor responsável pela orquestração de contêineres, utilizando o componente Heat a fim de orquestrar imagens de sistemas operacionais que

contém contêineres, e as executa em máquinas virtuais ou em máquinas físicas em configuração de *cluster*;

  - **Sahara:** é um *framework* de processamento de provisionamento de *big data*, com o objetivo de ofertar aos usuários uma maneira simples de provisionar *frameworks* de processamento de dados;
  - **Trove:** é um DBaaS (Banco de Dados como Serviço) capaz de provisionar motores de banco de dados relacionais ou não relacionais.

- **Ciclo de Vida de Aplicações:**
  - **Masakari:** é o serviço de instâncias de alta disponibilidade, tendo a função de recuperar automaticamente instâncias de máquinas virtuais com erro;
  - **Murano:** é o catálogo de aplicações do OpenStack, permitindo que desenvolvedores de aplicações e gestores de nuvem realizem a publicação de diversas aplicações prontas para a nuvem em um catálogo navegável. Desta forma, usuários da nuvem, incluindo usuários inexperientes, podem utilizar o catálogo a fim de configurar ambientes de aplicação de maneira rápida e fácil. Murano utiliza o componente Heat para realizar a orquestração dos recursos das aplicações do catálogo;
  - **Solum:** é uma ferramenta de automação de ciclo de vida do desenvolvimento de software, tornando os serviços de nuvem mais fáceis de serem consumidos e integrados, com os processos de desenvolvimento de aplicações através da automação dos processos de conversão de código para imagem conteinerizada, e simplificando a configuração de aplicações;
  - **Freezer:** é a ferramenta de gerenciamento de cópias de recuperação, restauração e recuperação de desastres do OpenStack. Ela foi projetada para ser compatível com diversos sistemas operacionais, e focar em prover flexibilidade e eficiência para cópias de recuperação em formatos baseados em blocos ou em sistemas de arquivo, ações *point-in-time*, sincronização de tarefas entre outras funcionalidades.

- ***Proxies* para APIs:**
  - **EC2API:** é a ferramenta que oferece APIs compatíveis com o Amazon EC2 [59][37] e o componente Nova.

- **Interfaces Web:**
  - **Horizon:** é o *dashboard* principal do OpenStack, oferecendo uma interface para monitoração e controle de todos os serviços do OpenStack.

3.2.2 OpenNebula

A plataforma de gerenciamento de nuvem OpenNebula [8] é uma das principais plataformas de gerenciamento de nuvem utilizadas atualmente [88]. Apesar de ser uma plataforma versátil, capaz de realizar o gerenciamento de nuvens publicas, privadas e híbridas, o seu foco maior são nos ambientes de nuvem privada, colocando as necessidades do cliente como ponto focal [86]. O OpenNebula foi projetado com o intuito de ser uma plataforma flexível, com baixo acoplamento e de fácil instalação, a fim de se adaptar às necessidades de uma grande quantidades de ambientes de nuvem.

Devido ao baixo acoplamento da plataforma, o OpenNebula oferece uma quantidade bem menor de recursos em comparação com o OpenStack, no entanto, o OpenNebula é compatível com diversas ferramentas, inclusive algumas projetadas para o OpenStack, para auxiliar na configuração e gerenciamento do ambiente em nuvem. Pode-se pegar como exemplo a utilização de ferramentas como o Open vSwitch [100] para criação de redes, roteadores vituais e firewalls; e o OpenStack Swift [6] para armazenamento de objetos ou dados do tipo blob [86].

O OpenNebula possui uma arquitetura simples, no entanto, apesar da simplicidade, a plataforma contém uma diversidade de recursos. Ela é compatível com diversas ferramentas externas, e está pronta para uso em infraestruturas homogêneas no ambiente corporativo. Além disso, o OpenNebula tem compatibilidade com os hipervisores VMWare, KVM e LXC e Firecracker [101].

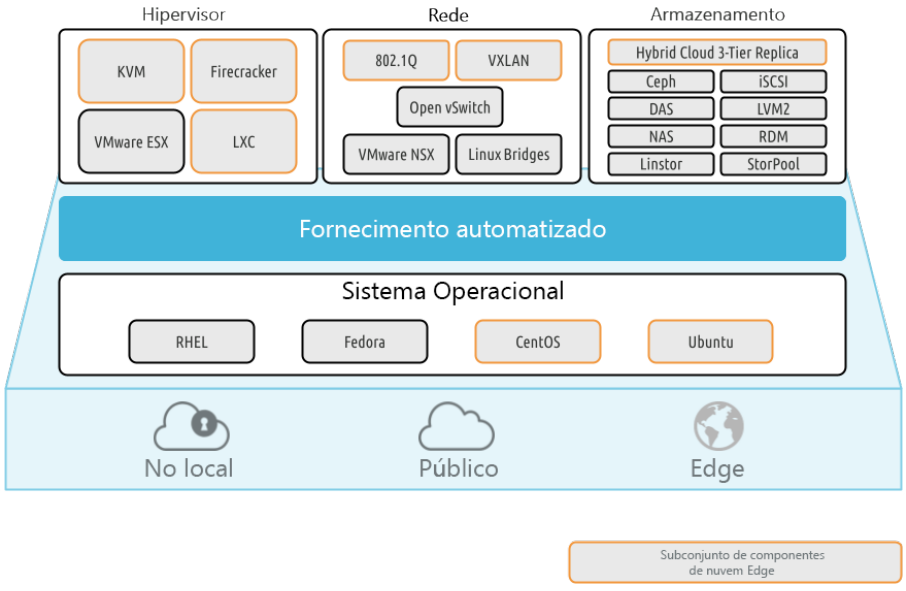

Figura 3.3: Arquitetura de nuvem tipo *Edge* da plataforma OpenNebula [7].

A arquitetura padrão do OpenNebula consiste do *Cluster* de Gerenciamento de Nuvem, junto com um nó do *cluster* responsável pela inteface *front-end*, e do restante da infraestrutura de nuvem, constituída do restante dos *clusters*. Cada *cluster* pode estar localizado em diferentes posições geográficas, configurações e tecnologias sem necessidade de centralizar a infraestrutura de nuvem [8].

Existem diferentes arquiteturas de *cluster* utilizadas pelo OpenNebula, a principal delas é o *Edge Cluster*, como apresentado na Figura 3.3, projetado para ser implantado em nuvens em públicas e nuvens privadas *on premise*, a fim de implementar verdadeiras nuvens híbridas [8]. Além disso, a arquitetura de nuvem *Edge* fornece uma solução mais simples, de fácil implantação, sem necessidade de uma configuração complexa, com o objetivo de prover um ambiente de nuvem híbrida sob demanda, com fácil integração com diversos hipervisores, recursos físicos e virtuais, e provedores de nuvem [8].

Em contrapartida, o OpenNebula também oferece um tipo de arquitetura de *cluster* mais customizável, na qual é necessária uma configuração mais complexa do ambiente. Mas oferece ao usuário um leque maior de opções, permitindo que o OpenNebula faça o gerenciamento de nuvem composta por diversas combinações de hipervisores, unidades de armazenamento e tecnologias de rede, como mostrado na Figura 3.4 [8]. Neste tipo de configuração, o usuário deve primeiramente configurar todos os componentes de software da infraestrutura em nuvem, e depois assim fazer a instalação do OpenNebula para a construção da infraestrutura de nuvem em si.

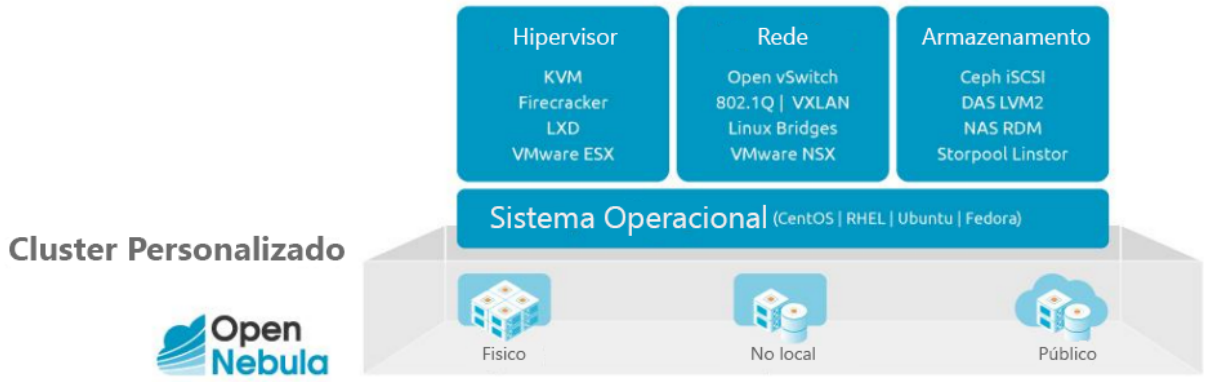

Figura 3.4: Arquitetura de *cluster* customizado do OpenNebula [8].

Não obstante, o OpenNebula também tem suporte a diversas ferramentas e *plugins*, a fim de auxiliar na realização de operações em imagens de máquinas virtuais, aplicações conteinerizadas e micro-VMs, como mostrado na Figura 3.5 [8]. Similarmente ao OpenStack, o OpenNebula utiliza-se de um componente *scheduler* a fim de realizar os

posicionamentos das máquinas virtuais, de acordo com a disponibilidade de recursos e o custo de execução [86].

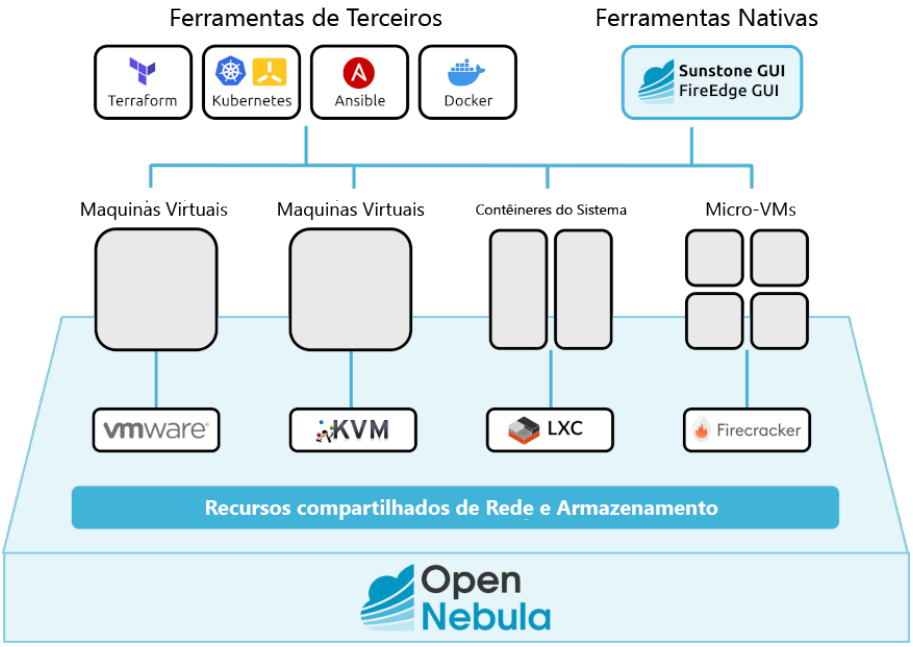

Figura 3.5: Visão geral da arquitetura do OpenNebula e seus componentes [8].

3.2.3 CloudStack

O Apache CloudStack [9, 102] é uma plataforma de gerenciamento de nuvem voltada para o gerenciamento e implantação de grandes redes de máquinas virtuais, com alta escalabilidade e disponibilidade no modelo IaaS (*Internet as a Service*). O projeto CloudStack começou em 2008 pela empresa *start-up* VMOps, depois chamada de Cloud.com. Desde o início do projeto haviam planos de tornar o código aberto ao público, visto que em 2010, uma parcela do código foi disponibilizado publicamente sob a *GNU General Public License version 3* (GPLv3). Após isso, em 2011 a Cloud.com foi comprada pela empresa Citrix que tornou o restante do código do CloudStack domínio público, novamente sob a GPLv3, lançando assim a versão 3.0 do Apache CloudStack, agora como um projeto totalmente em código aberto.

Atualmente, o CloudStack tem suporte para diversos hipervisores como VMWare, KVM, Citrix XenServer, Xen Cloud Plataform (XCP), Oracle VM Server e Microsoft Hyper-V. Além disso, o CloudStack contém uma diversa biblioteca de funcionalidades para auxiliar a implantação de um ambiente IaaS, como orquestração computacional, Rede Como Serviço (NaaS – *Network as a Service*), gerenciamento de contas de usuário, APIs nativas, controle de recursos e uma interface de usuário para configuração, monitoração

e gerenciamento do ambiente [102]. Adicionalmente, o CloudStack também oferece uma interface *web*, ferramentas de linhas de comando e uma API REST completa para controle da plataforma, como apresentado na Figura 3.6. A plataforma também oferece uma API compatível com o AWS EC2 e S3 para a implantação de nuvens híbridas.

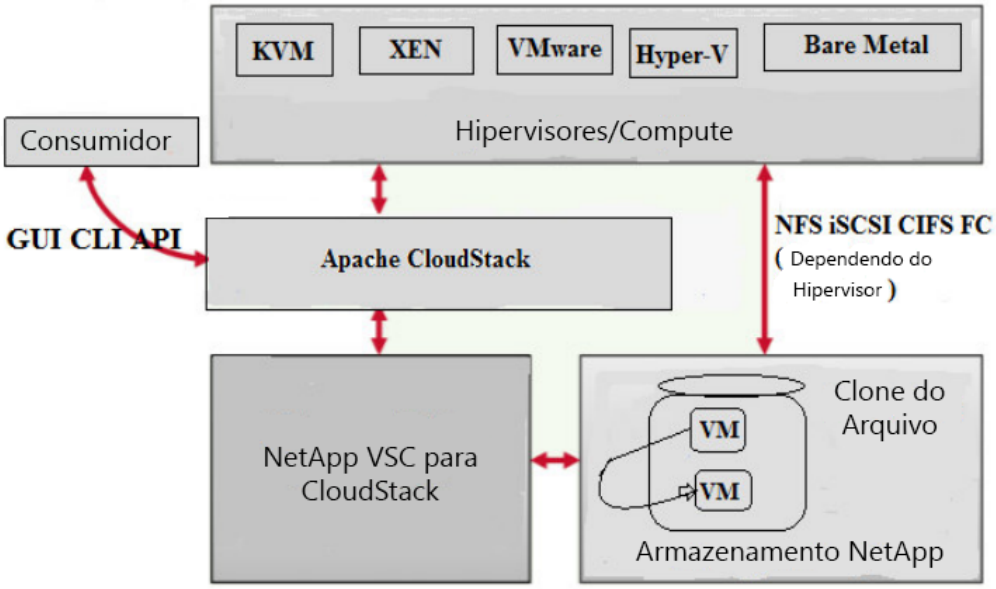

Figura 3.6: Arquitetura do CloudStack.

A arquitetura do CloudStack divide os elementos que compõem o ambiente de nuvem em diversos conceitos [102]:

- **Zona:** geralmente equivale a um único *datacenter*. Uma zona é constituída por um ou mais *pods* e pelo menos uma unidade de armazenamento secundário;
- ***Pod*:** geralmente equivale a um *rack* de hardware presente no *datacenter*, contendo um *switch* de Camada 2 e, pelo menos, um *cluster* de unidades computacionais;
- ***Cluster*:** conjunto de um ou mais *hosts* e pelo menos uma unidade de armazenamento primário;
- ***Host*:** unidade computacional dentro de um *cluster*, geralmente tomando forma de máquinas virtuais. É nesta unidade que é feito o trabalho computacional do ambiente em nuvem;
- **Armazenamento primário:** presente em um *cluster* e tem a função de armazenar os volumes de disco utilizados pelos *hosts* presentes no *cluster*;

- **Armazenamento secundário:** associado a uma zona e tem a função de fazer o armazenamento de imagens ISO, modelos e imagens de volume de disco instantâneos (*snapshots*).

Apesar de um *datacenter*, geralmente, representar apenas uma zona, existem casos em que há a necessidade de um *datacenter* fazer o papel de mais de uma zona. Neste caso, o *datacenter* passa a fazer o papel de uma Região, como mostrado na Figura 3.7 [9].

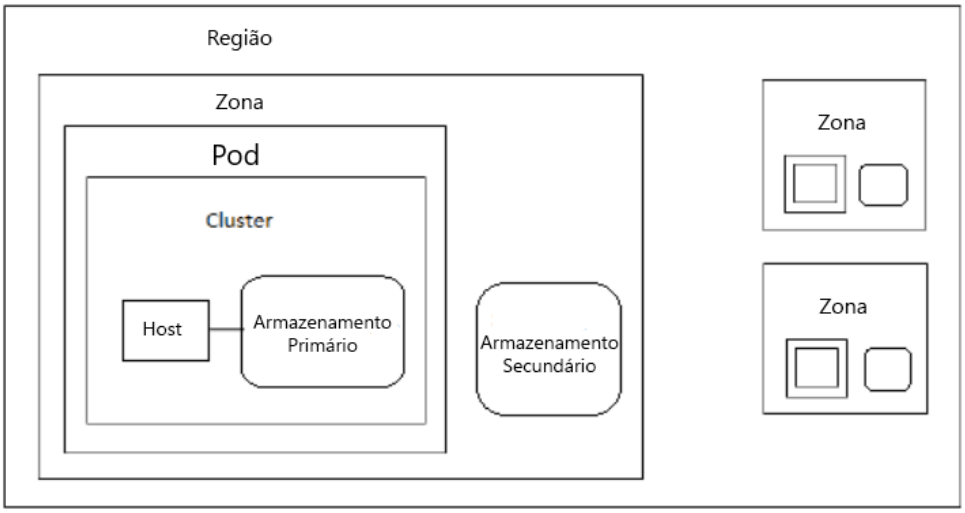

Figura 3.7: Exemplo de uma região com múltiplas zonas [9].

É comum em ambientes de nuvem implantados por meio do CloudStack, os elementos físicos da infraestrutura terem uma representação direta na arquitetura. Normalmente, um *datacenter* é representado por uma zona, um *rack* de máquinas é representado por um *pod* e um *cluster* é um grupo de máquinas dentro de um *pod* [102].

A gerência das máquinas e recursos em um ambiente de nuvem implantado pelo CloudStack é feita através de uma máquina de estados que controla o estado de cada recurso gerenciado, tais como máquinas virtuais, volumes, interfaces de rede e operações com *snapshots*. Em outras palavras, qualquer mudança de estado em alguns dos recursos gerenciados, que é feita por um usuário ou por algum gatilho pré-programado, gera um evento de mudança de estado que é lido e executado pela máquina de estados [102].

Como forma de organizar e gerenciar os *logs* de eventos, tanto síncronos quanto assíncronos, o CloudStack utiliza um Servidor de Gerenciamento que implementa uma abstração de barramento. Desta forma, qualquer alteração de estado de qualquer recurso gerenciado gera um evento de mudança de estado, no qual é publicado na máquina de estado correspondente ao recurso cujo estado está sendo modificado. Uma das maneiras de implementar o barramento de *logs* no Servidor de Gerenciamento é utilizando *plugins*

de integração com ferramentas de mensageria, como o RabbitMQ [103], que utiliza o protocolo AMQP (*Advanced Message Queuing Protocol*).

3.2.4 Eucalyptus

O Eucalyptus (*Elastic Utility Computing Architecture*) [104] é uma plataforma de gerenciamento de nuvem capaz de fazer a implantação e o gerenciamento de nuvens privadas e híbridas no modelo IaaS. Uma das suas características mais notáveis foi ser a primeira plataforma de gerenciamento de nuvem capaz de ser integrada com as APIs da AWS, tornando possível o gerenciamento de instâncias na nuvem privada e na nuvem pública da AWS, combinando-as em uma nuvem híbrida. A plataforma Eucalyptus é constituída dos seguintes componentes:

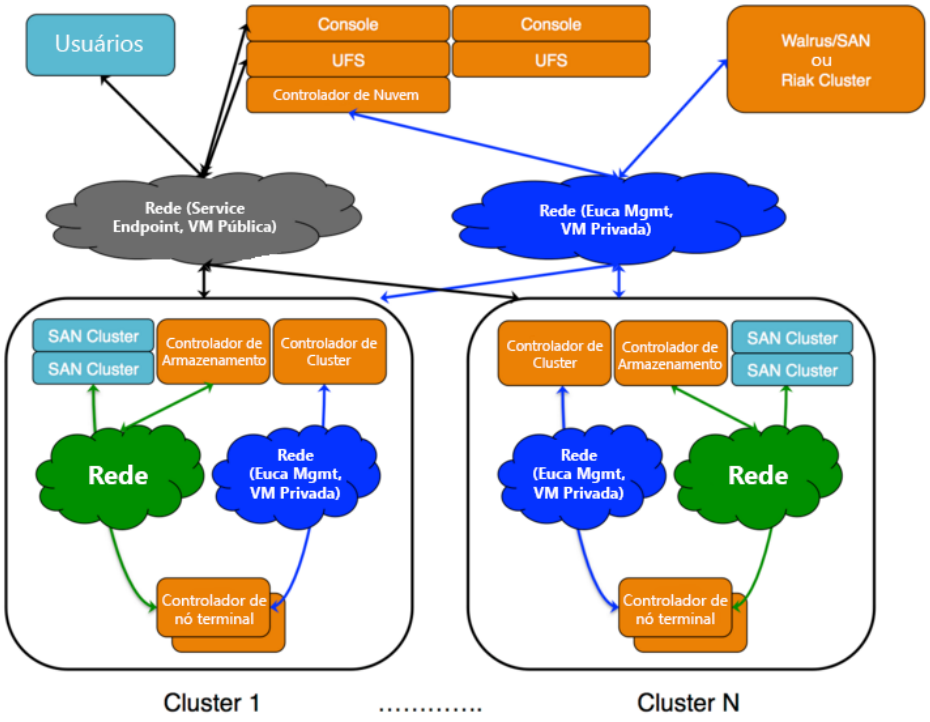

Figura 3.8: Arquitetura da infraestrutura de nuvem do Eucalyptus [10].

- **Imagem:** é um arquivo ou objeto que contém informações e configurações a serem lidas por um hipervisor, para que a partir destas configurações seja criada uma máquina virtual;
- **Instância:** nome dado a uma imagem que foi colocada em uso. Quando o hipervisor lê uma imagem e a partir dela é criada uma VM que é colocada em uso. Esta

VM é chamada de instância. O *Cloud Controller* é responsável por decidir o local em que a imagem será executada, além de fazer todo o gerenciamento de rede e armazenamento da instância;

- **Endereçamento:** assim que uma instância é criada, é atribuído a ela um endereço de IP. Instâncias podem ter IPs púbicos e privados e o Eucalyptus oferece suporte para endereços elásticos, no qual o IP de uma instância pode ser modificado em tempo de execução;
- **Segurança:** instâncias podem ser agrupadas em grupos de segurança TCP/IP, a fim de facilitar a configuração de regras de *firewall*. A fim de manter a segurança entre instâncias, cada instância de Camada 2 é isolada, de forma que um usuário em uma instância não pode fazer alterações em uma instância vizinha, visto que isso seria uma violação do princípio de separação de inquilinos (*tenants*);
- **Rede:** o Eucalyptus oferece aos usuários e gestores de rede três modos de configuração de rede: o modo gerenciado, o modo sistema e o modo estático. No modo gerenciado, o Eucalyptus fica responsável por toda a configuração e gerenciamento de rede, como grupos de segurança e endereços de IP. No modo sistema, a ferramenta fica responsável apenas pela criação de uma interface de rede às instâncias, atribuição de um endereços MAC e após isso, utilizando uma *bridge*, a ferramenta faz a anexação da interface de rede da instância com a rede física. No entanto, a ferramenta não oferece endereços IP elásticos, grupos de segurança ou isolamento de instâncias. Por fim, no modo estático, a ferramenta é responsável apenas pela atribuição de endereços IP às instâncias, de tal forma que IPs elásticos, grupos de segurança e isolamento de instâncias não são suportados;
- **Controle de acesso:** cada usuário do Eucalyptus recebe uma identidade, e dois ou mais usuários podem ser agrupados em grupos de acesso a fim de facilitar e centralizar atribuições de gerenciamento.

O ponto forte do Eucalyptus é a sua agilidade e praticidade. Além de ter compatibilidade com o sistema de mensageria da AWS que é o *Simple Queue Service* (SQS). No entanto, a ferramenta possui uma baixa escalabilidade, especialmente, ao manipular filas de mensagens ou fluxos de trabalho. Além disso, o Eucalyptus não oferece suporte a eventos de ciclo de vida de máquinas virtuais ou recursos e ferramentas para automação para elasticidade, além de possuir limitações na realização de customização da ferramenta, visto que o Eucalytus faz uso de diversos módulos fechados.

O Eucalyptus possui uma arquitetura baseada em quatro principais componentes, como apresentado na Figura 3.8. Os quatro principais componentes são [10]:

Tabela 3.1: Caracteriticas das principais CMPs

Características	**OpenStack**	**OpenNebula**	**CloudStack**	**Eucalyptus**
APIs suportadas	AWS/OCCI	AWS/OCCI	AWS	AWS
Hipervisores Recomendados	Xen, KVM, VMware	Xen, KVM, VMware	Xen, KVM, VMware	Xen, KVM, VMware
Arquitetura	Baseada em Componentes	Fracamente Acoplada	Fortemente Acoplada	Fortemente Acoplada
Nuvem Híbrida	Não	Sim	Sim	Sim
Sistema de Mensageria	RabbitMQ	Não	RabbitMQ e Kafka	AWS SQS

- ***Cloud Controller* (CLC):** módulo de gerenciamento central no qual os usuários e os administradores podem acessar os recursos em nuvem;
- ***Cluster Controller* (CC):** módulo responsável pelo gerenciamento de rede. O *cluster controller* deve ser executado em uma instância, dentro do mesmo *cluster* gerenciado;
- ***Storage Controller* (SC):** módulo responsável por fornecer armazenamento em bloco para a rede;
- ***Node Control* (NC):** módulo responsável pela monitoração das atividades das instâncias, tais como execução, inicialização e encerramento de instâncias.

A Tabela 3.1 apresenta as principais características das plataformas de gerenciamento de nuvem descritas anteriormente:

3.3 Desafios da Implementação de Nuvens Comunitárias no Modelo IaaS

Atualmente, uma das maiores dificuldades que grandes organizações com um maior tempo de mercado possuem ao implantar nuvens comunitárias é a integração com software legado. Isso adiciona uma complexidade extra, além de restringir diversas possibilidades de adaptação ao modelo de nuvem [26]. No entanto, é possível realizar uma mitigação preventiva deste problema, como o desenvolvimento e a manutenção de código, e imposição de políticas assertivas na operação de infraestrutura. Tais ações reduzem o impacto negativo que uma infraestrutura legada tem na migração para uma infraestrutura de nuvem moderna, visto que softwares utilizados se tornam mais previsíveis e de mais fácil adaptação a ambientes em nuvem [105]. Por outro lado, existem autores que defendem a ideia de que o principal desafio reside na capacidade dos provedores oferecerem o acesso uniforme aos serviços essenciais de nuvem, por meio de interfaces padronizadas, apesar de haver uma resistência na migração do controle de ambientes virtuais para plataformas de gerenciamento de nuvem [106].

Assim, observa-se que apesar dos diversos benefícios que potencialmente poderiam ser obtidos ao federar nuvens privadas em uma nuvem comunitária, o esforço da implementação, especialmente, em infraestruturas legadas que utilizam softwares que usam tecnologias defasadas, pode se tornar um empecilho considerável ao realizar a migração para uma arquitetura de nuvem comunitária [26, 105]. É necessário que seja feita uma análise de cada caso antes de ser determinada a melhor solução ao migrar para uma arquitetura de nuvem comunitária, pois cada ambiente possui características únicas de gerenciamento, configuração e administração. Além disso, é importante observar que além das limitações técnicas, existem também limitações negociais e burocráticas que devem ser levadas em conta a fim de estabelecer um acordo entre todas os entes da comunidade.

Dessa forma, no estudo de caso apresentado neste trabalho, órgãos do Poder Judiciário da União (PJU) utilizam softwares legados, possuem uma variedade de plataformas de virtualização e mecanismos de controle, e por este motivo, existe uma grande dificuldade em realizar a migração para uma arquitetura de nuvem moderna. Todavia, apesar da diversidade de ferramentas e soluções de gerenciamento e implantação de nuvem apresentadas anteriormente, observa-se que em sua grande maioria, estas soluções consideram recursos estáticos e pouco heterogêneos, de tal forma que não é possível obter um alto grau de customização e, consequentemente, ser impossível aplicar tais soluções para certos casos específicos.

Não obstante, analisando detalhadamente o caso do PJU, observa-se que se trata de uma estrutura dinâmica, com a utilização de diversos hipervisores e uma administração de infraestrutura altamente fluida, na qual grupos de acesso e controles de usuário precisam ser modificados de maneira rápida e em tempo de execução. Assim sendo, foi necessário pesquisar uma solução que seja capaz de ser integrada a este ambiente heterogêneo e dinâmico, a fim de prover um modelo de nuvem IaaS, levando em conta fatores como monitoração, segurança e controle de acesso de usuário. A solução encontrada foi o desenvolvimento e a implementação de uma plataforma de gerenciamento em nuvem chamada Cloud.Jus [27], capaz de ser integrada com a infraestrutura legada utilizada pelo PJU, e prover aos usuários uma solução de nuvem no modelo IaaS.

Assim sendo, a fim de endereçar as necessidades e os requisitos necessários para desenvolver uma ferramenta capaz de fazer o provisionamento e a gerência de uma nuvem no modelo IaaS, capaz de ser integrada com a infraestrutura do PJU, foi estabelecido que a ferramenta deveria conter as seguintes características [11, 81, 27, 107]:

- **Automatização:** uma nuvem que faz parte da comunidade deve ser capaz de identificar nuvens membros da federação e seus recursos disponíveis de maneira transparente, de forma que seja possível reagir à mudanças de maneira automática;

- **Previsão de carga de aplicações:** a ferramenta deve ser capaz de prever, de alguma forma, o comportamento dos serviços e a carga de demandas de forma eficiente e dinâmica, a fim de que seja possível fazer o escalonamento e o balanceamento de carga de tarefas de forma eficiente entre os membros;
- **Mapeamento de serviços e recursos:** deve ser feito o mapeamento entre os serviços e os recursos disponíveis de forma flexível, levando em conta o melhor custo-benefício e a garantia da qualidade de serviço e SLA;
- **Modelo de segurança interoperável:** deve ser permitida a integração de diferentes políticas de gerenciamento e tecnologias de segurança, sem que seja necessária a alteração das políticas internas dos membros da federação ao se aderir à federação de nuvem;
- **Escalabilidade no monitoramento de componentes:** a federação deve ser capaz de lidar com o volume de requisições de forma consistente, sem perda de desempenho, independente da quantidade de participantes da federação.

Diante disso, esse trabalho propõe a atualização da plataforma de gerenciamento de nuvem Cloud.Jus, utilizando ferramentas que auxiliem na interoperalização de componentes. Desta forma, cada componente da plataforma Cloud.Jus será mais modular e desacoplado, facilitando a implantação da plataforma em diferentes ambientes.

3.4 Trabalhos Relacionados

No âmbito acadêmico existem diversos trabalhos sobre o tema de CMPs e suas diversas implementações, contrapondo alternativas existentes, e recomendando soluções para usos específicos [108, 109, 110]. No trabalho de Couto *et al.* [111], é proposta uma solução em nuvem PID (*Plataform for IaaS Distribution*) a fim de prover um serviço IaaS para instituições de pesquisa, de forma que seja necessário o mínimo de gerenciamento e taxas de assinatura. Utilizando uma versão modificada do OpenStack, a fim de acomodar uma infraestrutura geo-distribuída, o PID utiliza-se do poder computacional dos servidores das instituições participantes da nuvem comunitária. Testes realizados no protótipo implementado em três universidades do Rio de Janeiro mostram bons resultados, visto que apesar da distância física entre os servidores, a latência de comunicação entre os servidores mostrou ter um impacto insignificante no controle de rede do OpenStack.

Não obstante, CMPs também vêm se provando ferramentas úteis para realizar o provisonamento em ambientes de computação de alto desempenho (*High Performance Computing* – HPC), como é o cenário descrito no trabalho de Bhardwaj *et al.* [90], no qual

é apresentada uma solução de nuvem privada chamada Megh. Nessa solução é utilizada a plataforma OpenNebula para realizar o provisionamento em um ambiente HPC. Semelhantemente, Razavi *et al.* [112] descrevem em seu trabalho experimentos com cargas de trabalho interativas em um ambiente de HCI, utilizando a elasticidade da nuvem a fim de disponibilizar recursos sob demanda para os usuários da plataforma, provisionando máquinas virtuais em tempo de execução.

Assim, é possível afirmar que, com base nos trabalhos dos autores mencionados, é necessário que durante o desenvolvimento de CMPs haja um foco grande na escalabilidade e elasticidade destas plataformas, a fim de tornar possível a integração com dezenas ou até mesmo centenas de servidores agrupados em *clusters*, e que seja possível armazenar e executar milhares de serviços, máquinas virtuais (VMs) e aplicações. Dessa forma, a eficiência de um sistema em nuvem depende das ferramentas, forma de implementação e configuração do sistema como um todo.

Logo, fica a cargo do administrador a escolha das ferramentas de gerência e a sua configuração, levando em conta a compatibilidade e dificuldade de integração com as outras partes do sistema, função e necessidade do serviço, a fim de se adequar às necessidades locais e suportar a carga de trabalho. Deve-se atentar que conforme a complexidade dos sistemas em nuvem aumenta, é necessária uma atenção e manutenção cada vez maior, de tal forma que a atenção e cuidado com os sistemas de monitoramento se tornam cada vez mais críticos, levando em conta o aumento da dispersão geográfica e a heterogeneidade do sistema [113].

Sob o mesmo ponto de vista, observa-se a crucialidade da monitoração dos sistemas em nuvem no trabalho de Cunha Rodrigues *et al.* [114], no qual é descrita a importância crescente dos sistemas de monitoramento e a necessidade da melhoria contínua de tais ferramentas, a fim de que seja possível a adaptação dinâmica do sistema de monitoração independente da escala e complexidade do sistema em nuvem. Desta forma, conforme os avanços na tecnologia de monitoração dos sistemas em nuvem vão ocorrendo, as ferramentas de monitoração vão se tornando cada vez mais abrangentes, autônomas e de fácil configuração e utilização. Observa-se também o foco dado para os critérios de capacidade de automatização, precisão dos dados de monitoração, métricas a serem monitoradas e até mesmo detecção e medição do impacto de migração de VMs, serviços ou aplicações para o sistema como um todo, ou até mesmo para o SLA estabelecido.

Por fim, conclui-se que a importância e a difusão de CMPs vem crescendo cada dia mais em instituições de médio a grande porte, especialmente, para a gerência interna de sistemas IaaS [24]. No entanto, para que seja feito um bom proveito das plataformas de gerenciamento de nuvem é necessário que haja muito cuidado, a fim de se fazer as escolhas corretas das ferramentas utilizadas. É necessário atenção à visão do sistema como um

todo, para que a combinação dos elementos do sistema funcionem de forma eficiente, provendo aos usuários os recursos necessários de maneira homogênea, independente do tipo de plataformas de infraestrutura utilizadas [87].

3.5 Considerações Finais

Neste capítulo foram apresentadas a definição de Plataformas de Gerenciamento de Nuvem (CMPs – *Cloud Management Plataforms*) e as principais CMPs disponíveis atualmente no mercado. Além disso, também foram abordados os principais desafios da implantação de nuvens comunitárias e exemplos de implantações bem-sucedidas de CMPs em diversas instituições, com diferentes necessidades. Isto foi necessário para introduzir as principais noções de CMPs, visto que o foco deste trabalho é a apresentação de uma plataforma de gerenciamento de nuvem para auxiliar na gerência de nuvem dos diferentes órgãos do PJU.

Assim, no próximo capítulo serão abordadas as principais ferramentas e tecnologias necessárias para o bom funcionamento desta plataforma, com o foco nos sistemas de virtualização utilizados e suportados pela plataforma. Além disso, serão abordados os serviços de mensageria implementados na plataforma, a fim de melhorar seu desempenho e torna-la mais modular e desacoplada.

Capítulo 4

Ferramentas e Tecnologias em Nuvem

A fim de modernizar a implementação da Plataforma de Gerenciamento Cloud.Jus, proposto por Castro *et al.* [27], este trabalho desenvolveu uma implementação mais robusta e eficiente da plataforma citada. Assim sendo, neste capítulo serão apresentadas tecnologias e ferramentas utilizadas na implantação de ambientes em nuvem. Na Seção 4.1 são apresentados noções e exemplos de sistemas de virtualização. Na Seção 4.2 o foco será em apresentar sistemas modernos de mensageria. Por fim, na Seção 4.3 serão apresentadas as considerações finais deste capítulo.

4.1 Sistemas de Virtualização

Dentre as opções de implementação de ambientes de nuvem apresentadas no Capítulo 2, tem-se que as nuvens públicas se mostram como uma das opções mais práticas e fáceis de se se implantar em qualquer ambiente, visto que a maioria da carga de trabalho e dificuldade de implantação fica a cargo do provedor. No entanto, em diversos cenários, não é interessante a uma empresa tornar toda sua infraestrutura em nuvem dependente de provedores públicos, pois pode existir dificuldade em executar uma possível troca de provedor, e também um possível impacto considerável no funcionamento do ambiente de nuvem da empresa [115]. Tal fenômeno é conhecido como *vendor lock-in* [116] e acontece quando uma instituição se vê bloqueada em algum provedor específico por estar dependente da estrutura de nuvem provida, limitando o controle que a instituição tem sobre sua infraestrutura de nuvem e dos custos e opções de hospedagem.

Adicionalmente, existem diversas instituições, como o PJU, que já possuem recursos proprietários que seriam desperdiçados caso a utilização de uma nuvem pública fosse a única opção disponível para que a instituição pudesse usufruir das vantagens de uma

infraestrutura em nuvem. Além disso, a utilização exclusiva de um provedor público pode causar dependência entre a instituição e as tecnologias proprietárias do CSP, como interfaces para o gerenciamento de recursos e serviços [117]. Outro fator que dificulta a mudança de provedores de soluções em nuvem é o tempo de adaptação e aprendizado de como utilizar as ferramentas de um novo provedor, e como realizar a gerência dos novos recursos, levando também em conta as diferenças entre os SLAs dos provedores.

Diante disso, observa-se que uma das soluções mais utilizadas por instituições com recursos proprietários *on-premises* é a nuvem híbrida [118], mostrando que as instituições estão buscando a virtualização de servidores como elemento-chave para viabilizar o provisionamento de ambientes em nuvem escaláveis, confiáveis e flexíveis para a hospedagem de aplicações. Segundo Sahoo *et al.* [119], a virtualização de servidores consiste na criação de máquinas virtuais (*virtual machines* ou VMs) que são abstrações do hardware subjacente, de forma que cada instância de máquina virtual possui um Sistema Operacional (SO), e cada instância é gerenciada por um sistema chamado hipervisor. Desta forma, é possível desacoplar os mecanismos de acesso dos recursos do servidor, facilitando a utilização, visto que torna possível a agregação de diversos sistemas heterogêneos no mesmo ambiente, como mostrado na Figura 4.1.

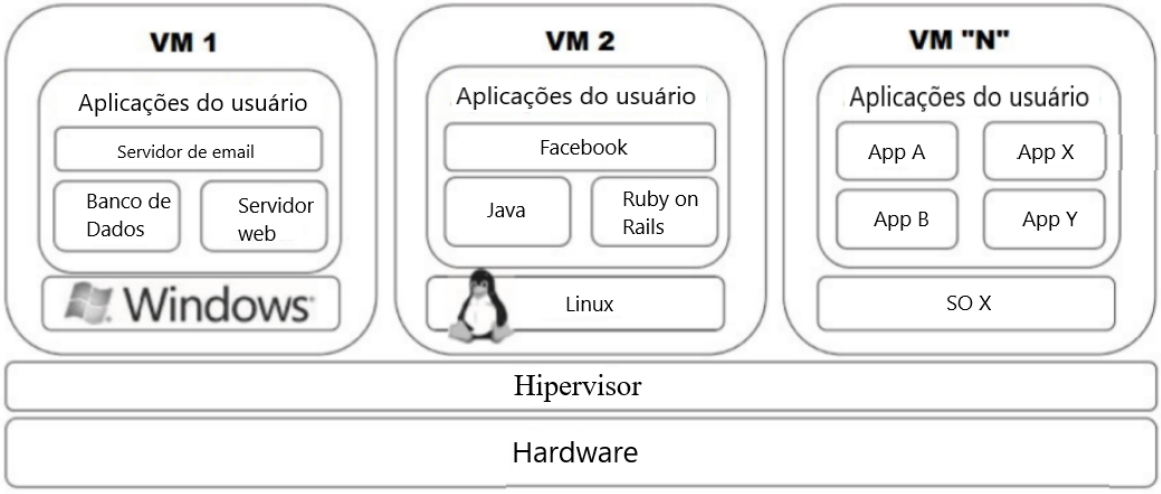

Figura 4.1: Arquitetura de servidores virtualizados, segundo Buyya [11].

A fim de iniciar a transição de uma infraestrutura legada para uma infraestrutura em nuvem, o desenvolvimento da plataforma Cloud.Jus teve seu início em 2017, com o intuito de modernizar e facilitar a gerência da nuvem privada do PJU [27]. Assim sendo, foram escolhidos os hipervisores Hyper-V [98], da Microsoft e VMware [97] para realizar a virtualização dos servidores proprietários do PJU, pois o mercado de virtualização de

servidores era liderado por estas duas empresas, segundo o "Relatório Quadrante Mágico 2016" do Instituto Gartner, apresentado na Figura 4.2 [50, 52].

Figura 4.2: Relatório Quadrante Mágico de Gartner 2016 para competidores no mercado de Virtualização de Plataformas x86 [12].

Apesar de ter uma participação grande do mercado de virtualização de servidores, a VMware é uma das empresas que vêm perdendo espaço no mercado de tecnologias em nuvem pelo grande crescimento de alternativas de hospedagem de nuvens públicas [50]. Adicionalmente, a VMware também divide espaço de mercado com as soluções fornecidas pela Microsoft, em especial, a solução de virtualização Hyper-V [98]. No entanto, Bittman [50] defende que, apesar de a plataforma VMware oferecer mais recursos em comparação à sua competidora, a plataforma Hyper-V, a Microsoft vêm oferecendo um produto sólido, com um preço mais baixo do que o seu concorrente. Assim sendo, a plataforma Hyper-V tem ganhado espaço no mercado, mantendo sua posição como segunda opção mais comum nos ambientes corporativos. Além disso, o autor também ressalta que a Microsoft vem implementando configurações de gerenciamento similares à sua nuvem pública, a Microsoft Azure [63], atraindo mais empresas interessadas em gerenciar e virtualizar seus ambientes locais com o Hyper-V. Não obstante, a Microsoft tem implementado configurações similares a outros produtos do seu catálogo, como é o caso da Azure Stack [120], oferecendo integração à toda pilha de infraestrutura e suporte a contêineres que

utilizam o sistema operacional Windows Server. Isso tem atraído clientes que necessitam implementar ambientes de desenvolvimento ou oferecer acesso remoto a usuários.

Por outro lado, outras formas de virtualização também vêm ganhando espaço no mercado, oferecendo a possibilidade de virtualizar outros componentes de infraestrutura, tais como sistemas, armazenamento e rede [119]. Pode-se tomar como exemplo a tecnologia de virtualização em contêineres, a qual adiciona mais uma camada de abstração, encapsulando não só o sistema operacional a ser utilizado, como também a própria aplicação e bibliotecas utilizadas pela aplicação. Desta forma, é possível implantar aplicações de forma rápida, fácil e sem dependência de ambiente e arquitetura de baixo nível, visto que todos esses componentes já estão incluídos e abstraídos dentro do próprio contêiner, aumentando a portabilidade entre diferentes ambientes e infraestruturas [121].

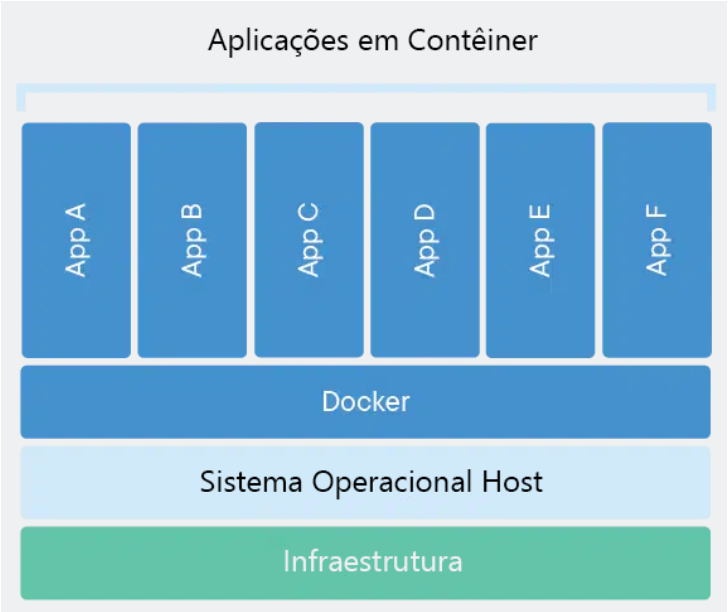

Figura 4.3: Arquitetura de aplicações baseadas em contêineres [13].

Na arquitetura de virtualização de contêineres, como mostra a Figura 4.3, o contêiner em si fica encarregado em fornecer todo o ambiente em que a aplicação ou as aplicações serão executadas, enquanto o gerenciamento da infraestrutura fica em cargo do provedor de computação em nuvem [13]. Desta forma, a virtualização em contêiner permite que uma aplicação seja instanciada e executada em outro ambiente, cujo SO é compatível com a ferramenta de virtualização de contêineres utilizada, permitindo a execução até mesmo entre sistemas operacionais distintos [122].

Atualmente, uma das ferramentas mais utilizadas na virtualização em contêineres é a ferramenta Docker [13], que popularizou a tecnologia e tornou-se a ferramenta padrão na implantação de serviços baseados em contêineres [13]. Com isso, tornou-se possível

a migração de aplicações para diferentes *datacenters* distribuídos, nem necessidade de configurar ambientes distintos compatíveis com diversas aplicações individuais, visto que todo o ambiente necessário para a execução da aplicação já está incluso dentro do contêiner [122].

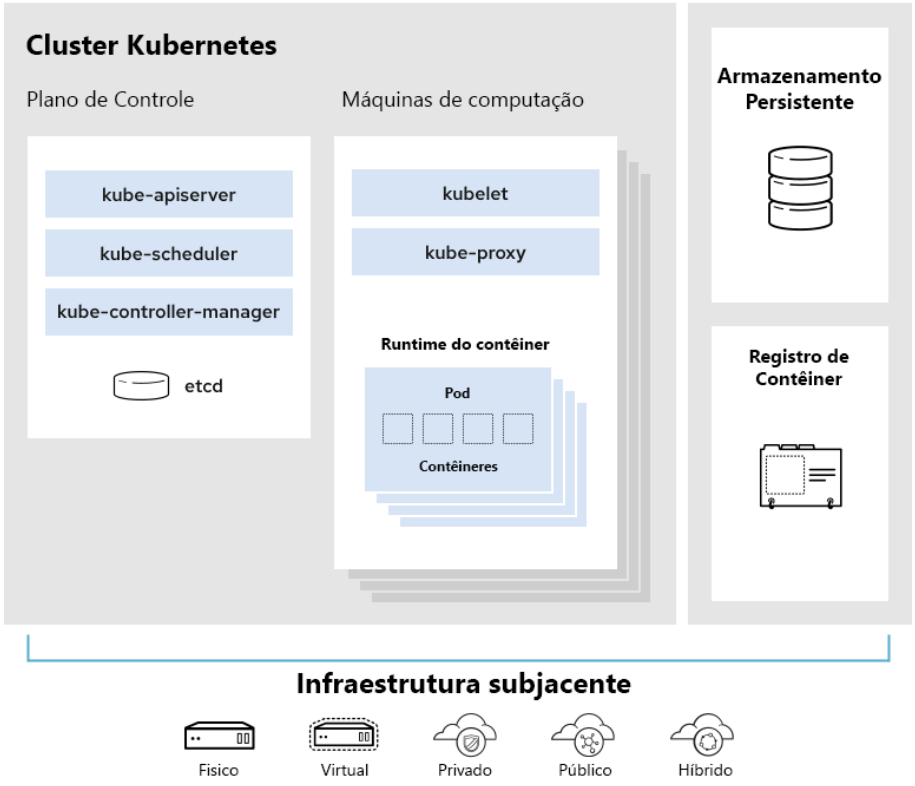

Figura 4.4: Arquitetura do orquestrador de contêineres Kubernetes [14].

Por outro lado, com a ascensão de tecnologia de virtualização baseada em contêineres, originou-se também plataformas com o intuito de realizar a orquestração e a automação para esse tipo de tecnologia. Ferramentas como o Kubernetes [14] e OpenShift [123] vêm ganhando espaço no mercado, oferecendo eficiência e uma melhor experiência ao gerenciar um ambiente que faz uso de aplicações conteinerizadas. Isso é vantajoso porque elimina grande parte da carga de processos manuais necessários na implantação e gerência de aplicações em nuvens privadas, públicas ou híbridas.

Assim sendo, tomando o orquestrador Kubernetes como exemplo, a Figura 4.4 mostra a arquitetura de um orquestrador de sistemas virtualizados em contêineres. Pode-se observar que o Kubernetes organiza contêineres em *pods*, no qual são executados em nós de um *cluster*. *Pods* são instâncias de aplicações ou serviços conteinerizados ou de máquinas virtuais [14]. A fim de realizar a comunicação com os *pods*, o Kubernetes utiliza um

servidor mestre, no qual o administrador ou uma equipe de DevOps [124], pode executar comandos que por sua vez são transmitidos para os nós em execução [14].

4.2 Sistemas de Mensageria Distribuída

Uma das ferramentas mais úteis na integração de componentes em nuvem é a *Application Programming Interface* (API) que torna possível realizar a comunicação entre ambientes e componentes distintos. Tal característica é importante, visto que viabiliza um dos principais requisitos para a realização da computação em nuvem, que é a sua capacidade de disponibilizar múltiplos recursos, independente do ambiente de infraestrutura e mantendo a independência de localidade [84]. Paralelamente, serviços de mensageria permitem a comunicação entre serviços e elementos de nuvem de forma desacoplada, gerenciando toda a carga de mensagens e requisições. Desta forma, os clientes podem focar seu trabalho na comunicação entre os serviços, sem se preocuparem ou lidarem diretamente com os terminais (*endpoints*) dos serviços [15]. Assim sendo, sistemas de mensageria permitem que os diversos componentes de nuvem, incluindo hardware e software, possam comunicar entre si, compartilhando dados ou coordenando ações, independente da sua localização, e sem afetar a experiência do usuário. Assim, os diferentes elementos podem trabalhar e se comunicar de forma conjunta como se fossem um sistema único [125].

Apesar da popularidade que a API REST possui atualmente, é possível aprimorar ainda mais a sua utilização caso seja utilizada em conjunto com sistemas de mensageria Pub/Sub (Publicação/Subscrição) [15], que já possuem a capacidade de serem integrados com sistemas heterogêneos, de forma escalável, a fim de implementar um sistema distribuído no modelo de nuvem. A fim de realizar a tarefa de integrar diversos componentes distintos e, diversas vezes, localizados fisicamente em diferentes locais, os sistemas de mensageria contam com três características específicas [126]:

- **Dissociação de entidades:** não é necessário que produtores e consumidores tenham conhecimento um do outro. É dever do serviço de mensageria determinar o início e o fim da comunicação entre produtores e consumidores;
- **Desacoplamento de tempo:** não é necessário que produtores e consumidores estejam ativos simultaneamente durante a comunicação;
- **Desacoplamento de sincronização:** não é necessário a sincronia entre produtores e consumidores. Logo, produtores e consumidores podem se comunicar de forma assíncrona, sem necessidade de ambas as partes estarem ativas simultaneamente durante a comunicação.

Visto isso, observa-se que sistemas de mensageria se tornam candidatos ideais para processamento *logs* e troca de mensagens, dando origem as ferramentas como RabbitMQ [103], Kafka [127] e NATS [128], os quais são ferramentas de mensageria com arquitetura distribuída. Nestes serviços, mensagens publicadas pelos publicadores são colocadas em filas de mensagens para serem consumidas pelos consumidores. Estas filas podem ser categorizadas de duas formas: filas intermediadas e não intermediadas. As filas intermediadas (ou *broker-based*) são filas que são gerenciadas por um servidor central que faz papel de um mediador entre produtores e consumidores; enquanto filas não intermediadas (ou *peer-to-peer*) são aquelas nais quais não há nenhum serviço intermediador entre produtores e consumidores. Adicionalmente, os serviços de mensageria podem adotar dois padrões básicos de comunicação: padrão solicitação/resposta (*request/reply*) ou RPC (*Remote Procedure Call*), utilizado para serviços; e o padrão de fluxo de evento de dados [129].

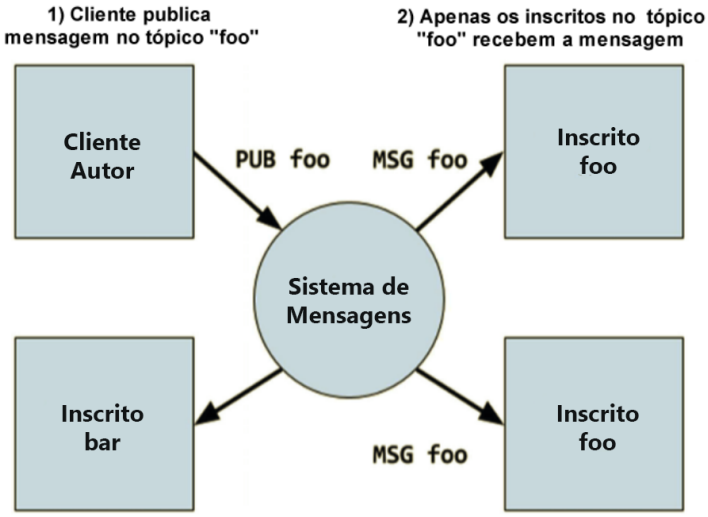

Figura 4.5: Arquitetura de Publicação/Subscrição do serviço de mensageria NATS [15, 16].

Uma das maiores vantagens de se utilizar um sistema de mensageria é que toda a carga de trabalho de implementação de mecanismos de tolerância a falhas e redundância fica sob supervisão do próprio sistema de mensageria, facilitando o desenvolvimento de aplicações destinada a sistemas e ambientes de alta latência de rede, como em ambientes compostos por *datacenters* geo-distribuídos. Outra vantagem de serviços de mensageria é a sua fácil utilização e funcionamento, de forma que clientes fazem papel de consumidores, recebendo mensagens publicadas em canais ou tópicos específicos. Desta forma, sempre que um produtor publicar uma mensagem em um determinado canal ou tópico, todos os consumidores que estão "inscritos" naquele canal receberão a mensagem, como mostra a Figura 4.5.

4.2.1 RabbitMQ

O RabbitMQ [103] é um serviço de mensageria que utiliza o *Advanced Message Queuing Protocol* (AMQP) a fim de fazer o gerenciamento de diversos *middlewares* de mensagens assíncronas de forma eficiente e escalonável. O AMQP é, por sua vez, um protocolo de transporte de mensagens projetado com o foco em desempenho, escalabilidade e confiabilidade [126].

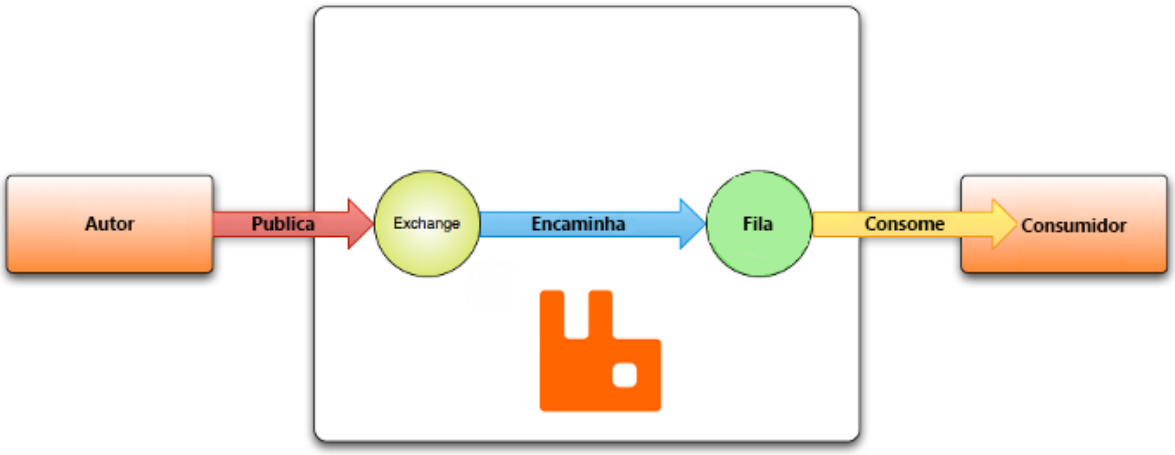

Figura 4.6: Arquitetura do serviço de mensageria RabbitMQ [17].

Apesar de implementar o AMQP, o RabbitMQ executa o controle de fluxo de forma mais eficiente, implementando um mecanismo transacional modificado. Além disso, o RabbitMQ também suporta outros protocolos de mensageria por meio de *plug-ins*, como *Streaming Text Oriented Protocol* (STOMP), para transmissão de texto; e o *Massage Queing Telemetry Transpor* (MQTT), utilizado em ambientes de alta latência de rede, com o foco em dispositivos IoT (*Internet of Things*). Como mostrado na Figura 4.6, é possível observar que o RabbitMQ utiliza uma arquitetura modular e desacoplada, dividindo o mecanismo mediador de mensagens em duas funções, troca e filas [17]. Uma troca funciona como um roteador, com a função de receber mensagens e roteá-las para a fila correta, segundo uma série de regras e configurações. A fila tem a função de armazenar as mensagens e enviá-las aos consumidores. As suas configurações de armazenamento, em disco ou em memória, e a capacidade podem ser configuradas pelo administrador do sistema.

4.2.2 Kafka

O Apache Kafka [127] é um sistema de mensageria que foi primeiramente projetado pela empresa LinkedIn com a função de auxiliar no transporte de mensagens de eventos

de aplicações da empresa [18]. O Kafka é um sistema de mensageria PubSub escalável, projetado para transmitir uma alta quantidade de mensagens, na ordem de bilhões, mostrando-se uma ferramenta extremamente útil, especialmente, em ambientes que lidam com um grande volume de dados [130].

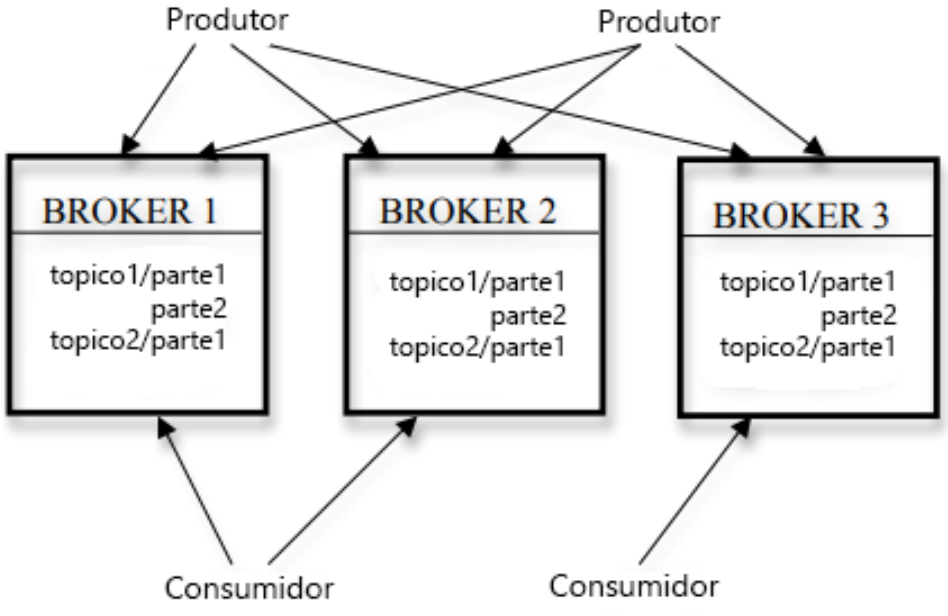

Figura 4.7: Arquitetura do sistema de mensageria Kafka [18].

Na arquitetura do sistema Kafka, produtores enviam mensagens para um determinado tópico, sendo que cada tópico funciona como um canal, no qual consumidores se "inscrevem" para receber as mensagens enviadas em um determinado tópico. Os tópicos são distribuídos entre um conjunto de *brokers*, no qual cada *broker* hospeda partições de cada tópico. Cada partição é efetivamente um *log* gravado em disco, o qual pode ser acessado pelos consumidores de forma que o peso da carga e transporte de mensagens e *logs* se mantém em sua maioria nos próprios *brokers*. Isso permite manter uma carga baixa nos consumidores e garantir a escalabilidade dos mesmos, como indicado na Figura 4.7 [18].

Adicionalmente, o Apache Kafka também utiliza uma ferramenta chamada Apache Zookeeper [131] para fazer o papel de controlador, a fim de otimizar o funcionamento do sistema. Para otimizar ao máximo o funcionamento do sistema de mensageria, o Apache Zookeeper utiliza técnicas tais como processamento em lote, estruturas de dados persistentes em cache e leitura antecipada do sistema operacional [126].

4.2.3 NATS

O NATS é um serviço de mensageria desenvolvido para realizar o barramento de mensagens da ferramenta Pivotal Cloud Foundry, uma solução com a função de realizar a orquestração de contêineres da Dell e VMware. No entanto, com o aumento da popularidade de soluções e aplicações que utilizam a arquitetura de microsserviços e ambientes em nuvem, o NATS se mostrou uma ferramenta poderosa para auxiliar na integração e comunicação dos elementos que compõem um ambiente em nuvem. Em especial, o NATS destacou-se ao fornecer à aplicações funcionalidades como [15]:

- Descoberta de serviços;
- Comunicação de baixa latência;
- Balanceamento de carga;
- Notificações e gerenciamento de eventos.

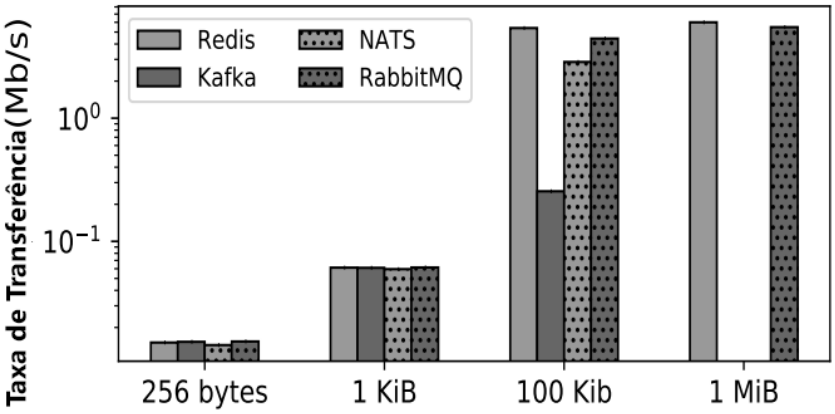

Figura 4.8: Comparação de vazão de mensagens entre diferentes servicos de mensageria [19].

Adicionalmente, o NATS se destaca entre outros serviços de mensageria por se tratar de uma solução que não depende de componentes adicionais, tais como o Zookeeper e JVM, que são necessários para realizar o instanciamento do Kafka [18], por exemplo. Além disso, o protocolo de mensageria utilizado pelo NATS é baseado em texto, o que torna possível receber e enviar mensagens através de conexões como Telnet, facilitando a comunicação entre aplicações executadas em diferentes sistemas operacionais. Ademais, *benchmarks* de solicitação e resposta demonstram um maior desempenho do NATS em relação à outros serviços de mensageria como o Kafka, ao lidar com pacotes de mensagens com um *payload* menor do que 1MB, como indicado na Figura 4.8 [15, 19].

Visto isso, o NATS foi a ferramenta escolhida para auxiliar no transporte e organização de filas da nova versão da Platafroma de Gerenciamento Cloud.Jus [27], pelo seu baixo acoplamento ou necessidade de componentes auxiliares, além da sua fácil integração por ser um serviço de mensageria baseado em texto. Na próxima Sessão deste trabalho será detalhada a implementação do NATS na plataforma Cloud.Jus.

4.3 Considerações Finais

Neste capítulo foram apresentadas diferentes ferramentas e tecnologias utilizadas durante o desenvolvimento e a implantação da nova versão da plataforma de gerenciamento de nuvem Cloud.Jus [27]. A plataforma Cloud.Jus foi desenvolvida com o suporte para tecnologias de virtualização como VMware [97] e Hyper-V [98], e este trabalho se propõe a apresentar uma atualização da plataforma, utilizando novas tecnologias como a conteinerização dos componentes da plataforma e utilizando ferramentas de mensageria como o NATS [128], para facilitar a interoperabilidade entre seus componentes.

Assim, no próximo capítulo será apresentada a nova arquitetura da plataforma Cloud.Jus. Para isso serão abordados os detalhes sobre as tecnologias, linguagens de programação, *frameworks* e os tipos de ferramentas de armazenamento utilizados no desenvolvimento da plataforma.

Capítulo 5

Plataforma de Gerenciamento em Nuvem Cloud.Jus3.0

Este capítulo apresenta a arquitetura da plataforma Cloud.Jus, além de apresentar as melhorias e os novos componentes implementados na plataforma. A Seção 5.1 apresenta a visão geral da plataforma. Na Seção 5.2 é feita a descrição conceitual do Cloud.Jus, enquanto a Seção 5.3 descreve as funcionalidades de nuvem que a plataforma oferece. Na Seção 5.4 é apresentado o painel de controle da plataforma, enquanto na Seção 5.5 é apresentado um dos novos componentes implementados na nova versão da plataforma, a *Application Programming Interface* (API). Na Seção 5.6 é apresentado o componente orquestrador (*middleware*) e as melhorias implementadas nele. Nas Seções 5.7, 5.8 e 5.9 são descritos como é realizado o gerenciamento de recursos, monitoramento e gerenciamento de eventos da plataforma, respectivamente. Por fim, a 5.10 aborda a avaliação experimental realizada a fim de comparar o desempenho da plataforma antes e após a implantação das melhorias propostas neste trabalho.

5.1 Visão Geral

A plataforma Cloud.Jus foi desenvolvida a fim de auxiliar na gerência do ambiente de nuvem do PJU. Durante o desenvolvimento da ferramenta foram tomadas certas precauções durante do desenvolvimento do projeto. Uma das precauções tomadas foi não desenvolver uma solução monolítica e fortemente centralizada, visto que tal abordagem é limitada em ambientes complexos, heterogêneos e mutáveis, como é o caso do ambiente e infraestrutura do PJU [26, 132]. Visto isso, buscou-se separar e modularizar cada um dos componentes que compõem a plataforma, de forma que seja possível o fácil acesso, manipulação e gerência de cada componente em específico. Outra vantagem de se utilizar um

projeto modular é a habilidade de isolar componentes problemáticos e realizar correções sem que haja o comprometimento dos demais componentes do sistema.

Outra característica importante que foi definida durante a fase de projeto do sistema, foi que não seria interessante criar uma ferramenta universal capaz de solucionar todos os problemas simultaneamente. Primeiramente, foi definido que a ferramenta deveria ser modular, de forma que cada componente seja desenvolvido individualmente, um por vez. Desta forma, o resultado final seria um sistema modular, com baixo acoplamento e com uma baixa complexidade geral [133]. Além disso, o foco do projeto inicial foi criar um sistema capaz de solucionar os problemas mais críticos do ambiente de nuvem do PJU, e, a cada iteração de desenvolvimento, aumentar o escopo do projeto.

Não obstante, o sistema deveria ser projetado e desenvolvido com o foco principal na velocidade ao invés de focar na completude do sistema. Segundo Edwards [20], é importante que as organizações vejam ferramentas com uma visão em cadeia, pois é um *design* no qual é possível padronizar para todos. Esta abordagem garante um baixo acoplamento, visto que ter boas ferramentas individuais é a forma mais fácil de realizar algo ou solucionar algum problema.

Considerando a quantidade de ferramentas já presentes no mercado e os recursos nativos presentes no ambiente do PJU, foi determinado que para facilitar o processo e poupar tempo, o sistema deverá tomar proveito das ferramentas e recursos já disponíveis, visto que tais ferramentas já estão maduras no mercado, em especial hipervisores corporativos [134]. Ao utilizar serviços já oferecidos pelos hipervisores, é possível poupar tempo valioso de desenvolvimento, visto que hipervisores de mercado são fáceis de configurar, além de já oferecerem APIs, consoles de gerenciamento e muitas vezes interfaces web para facilitar a gerência e monitoramento do ambiente. Adicionalmente, hipervisores são capazes de interagir com a infraestrutura base, facilitando a operação de ambiente e tornando desnecessário a manipulação direta dos recursos de infraestrutura.

Assim sendo, a plataforma Cloud.Jus foi projetada para trabalhar em conjunto com os hipervisores disponíveis no PJU, a fim de facilitar a gerência e a manipulação dos recursos em forma de infraestrutura como serviço em nuvem, não só por parte dos administradores, mas também para disponibilizar uma ferramenta intuitiva para que o próprio usuário final possa realizar o provisionamento de recursos necessários para o seu trabalho com segurança e controle. Além disso, O Cloud.Jus foi projetado para ser altamente modular e desacoplado, capaz de realizar o provisionamento de recursos independente do provedor e ambiente de nuvem.

A plataforma Cloud.Jus adiciona uma camada de abstração a mais, realizando a comunicação entre os hipervisores e o administrador e usuários finais do sistema. A utilizar a plataforma, o usuário é capaz de requisitar o provisionamento de máquinas virtuais,

manusear e acessar as VMs já existentes e obter uma visão geral do ambiente em nuvem. O controle de acesso é controlado por políticas de provisionamento determinadas pela própria organização. Tomando como exemplo o PJU, é utilizado o Active Directory da Microsoft [135] como provedor de identidade interno, a fim de separar usuários em grupos de acesso, no qual cada grupo contém um nível de acesso aos recursos em nuvem.

Desta forma, a plataforma Cloud.Jus foi criada com o intuito de prover um ambiente IaaS a organizações que buscam tomar proveito das vantagens de se utilizar uma arquitetura de computação em nuvem, mas possuem um ambiente que utiliza ferramentas legadas com alto acoplamento. O plataforma Cloud.Jus, busca realizar a comunicação com tais ferramentas, adicionando uma nova camada de abstração, centralizando o gerenciamento de recursos de infraestrutura e provendo um ambiente de nuvem do tipo IaaS para seus usuários finais.

Após a primeira implementação da plataforma, observou-se a necessidade e a oportunidade da realização de melhorias na plataforma como um todo. Assim sendo, foram propostos e implementados novos componentes e ferramentas para modernizar e aumentar a eficiência da plataforma. O foco deste capítulo é apresentar os principais conceitos e funcionalidades da plataforma. Após isso, serão apresentadas as melhorias implantadas, comparando a eficiência da plataforma antes e depois da implementação das melhorias propostas.

5.2 Descrição Conceitual

A fim de tornar possível a criação de uma plataforma de gerenciamento desacoplada e altamente modular, foi definido que a plataforma deveria ser capaz de tolerar a troca de fornecedor de seus componentes, serviços e processos [117, 5]. Além disso, a plataforma deve conter as seguintes características[136]:

- **Foco no Público Alvo:** as soluções e resultados devem ser planejados e realizados de acordo com a melhor experiência de usuário;
- **Encontrar boas soluções:** as soluções encontradas devem priorizar as necessidades do público alvo;
- **Prototipação ágil:** é interessante a utilização de modelos e projetos rápidos, de forma que seja possível descartar más soluções de forma rápida, sem um grande prejuízo de tempo;
- **Trabalho Colaborativo:** aqueles que agem e trabalham com a solução não devem trabalhar de forma isolada, visto que isso pode acarretar a perda de contexto e prejudicar a proposta do sistema;

- **Soluções Personalizadas:** cada solução desenhada deve ser adaptada ao ambiente e ao contexto de cada organização, levando em conta suas especificidades.

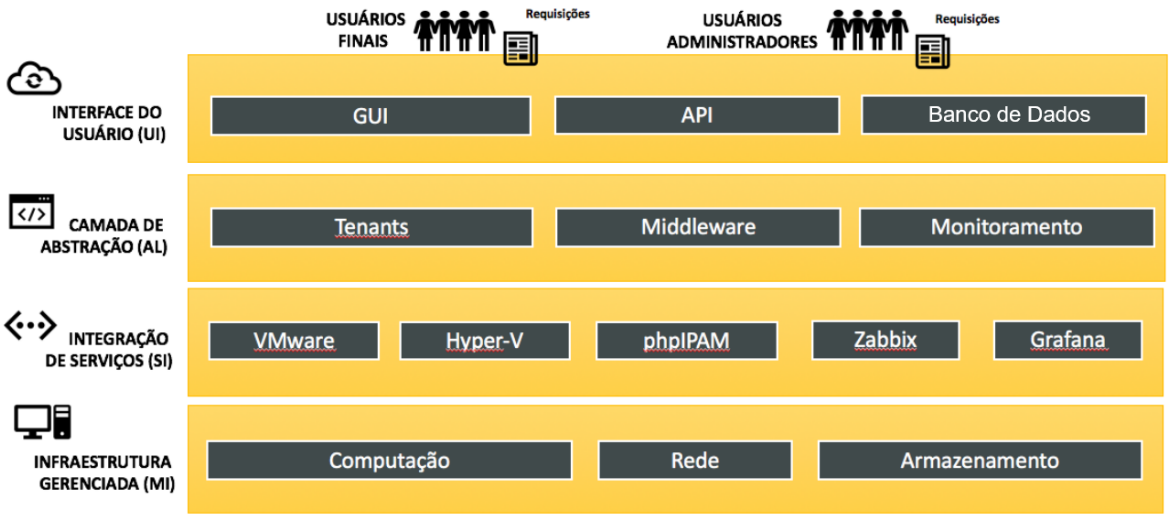

Figura 5.1: Visão geral da arquitetura da plataforma Cloud.Jus.

Na Figura 5.1 é mostrada a arquitetura de alto nível da plataforma de gerenciamento Cloud.Jus, mostrado em um abordagem *top-down*. Na figura, arquitetura da plataforma é dividida em 4 camadas: A Camada de Interface de Usuário (UI - *User Inteface*), a Camada de Abstração (AL - *Abstration Layer*), a Camada de Integração de Serviços (IS - *Integration Services*) e, por fim, a Camada de Infraestrutura Gerenciada (MI - *Managed Infrastructure*).

A Camada de UI é a camada na qual usuários e administradores terão acesso aos recursos em nuvem. A camada contem três componentes principais: Uma Inteface Gráfica de Usuário (GUI - *Graphic User Interface* [137], onde o usuário pode acessar os recursos de nuvem, acessar *dashboards* contendo informações sobre os recursos utilizados e realizar requisições de manipulação de recursos, como criação ou remoção de máquinas virtuais; Uma API de controle, responsável por realizar a comunicação entre a GUI e as camadas inferiores do sistema; e um banco de dados não-relacional responsável por armazenar informações sobre usuários, máquinas virtuais e controle de acesso.

A Camada de AL é responsável por realizar a comunicação entre a camada mais externa de interface de usuário com a camada mais interna de integração de serviços. Ao contrário da camada de interface, apenas administradores têm acesso à esta camada, e a sua função é estabelecer o conceito de *tenants*, cada *tenant* representa um conjunto de recursos pré-determinados, que podem ser instanciados por usuários da infraestrutura de nuvem; conter o *middleware* responsável por realizar a comunicação entre a API na

camada de interface com os hipervisores na camada de integração de serviços; e, por fim, realizar o monitoramento e controle de *logs* das atividades realizadas e recursos utilizados.

A Camada de SI contém as ferramentas que realizam o gerenciamento de recursos de infraestrutura da camada inferior. Nesta camada estão presentes os *clusters* dos hipervisores, regras de monitoramento de baixo nível e endereçamento de rede. Esta camada foi projetada para se utilizar de soluções nativas oferecidas pelas plataformas já presentes e realizar a integração destas soluções com as camadas de mais alto nível. A função principal da camada de SI realizar a leitura das requisições de usuários vindas das interfaces *front-end* e repassar estas instruções para as plataformas de gerenciamento de infraestrutura já instalados na organização.

A Camada MI é a camada de mais baixo nível do sistema. Nela são feitos as operações de manipulação direta dos recursos de infraestrutura, como servidores físicos, ativos de rede, unidades de armazenamento, *firewalls*, entre outros. A manipulação desta camada está fora do escopo deste projeto, e a responsabilidade do micro-gerenciamento dos recursos de infraestrutura recai sobre a organização proprietária de tais recursos.

O *design* proposto para a plataforma Cloud.Jus é um *design* com foco no cliente, administradores e na experiência do usuário ao contrário de propostas com o foco no provedor. Com isso, a plataforma é capaz de preservar mais facilmente a compatibilidade entre componentes até mesmo em ambientes heterogêneos, disponibilizando uma interface amigável para o usuário e fazendo uso de ferramentas suportadas por seus fabricantes, como módulos Powershell [138] para gerenciamento de hipervisores como VMware e Hyper-V.

5.3 Funcionalidades de Nuvem

Nesta seção são descritas as principais funcionalidades da nuvem implementadas na plataforma de gerenciamento Cloud.Jus:

- **Painel de Controle - *Dashboard* (GUI):** é a interface gráfica da plataforma de gerenciamento. É uma interface *web* escrita nas linguagens *PHP* [139] e HTML5 (*HyperText Markup Language*) [140]. Por meio dela, usuários e administradores são capazes de acessar e monitorar os recursos em nuvem. Possui um sistema de controle de acesso, no qual usuários são alocados em grupos de acessos com permissões específicas e configuráveis pelo administrador do sistema;
- ***Application Programming Interface* (API):** funciona como o *backend* para a interface gráfica. Atraves da API, a inteface gráfica pode enviar requisições para as camadas de abstração de mais baixo nível e receber informações sobre máquinas virtuais e alocação de recursos de nuvem. A API foi escrita utilizando a linguagem

Javascript [141] e utiliza o *framework* NodeJS [142] para receber requisições HTTP REST [143];

- **Banco de Dados:** para auxiliar no armazenamento de dados e informações de máquinas virtuais, *templates* de VMs, controle de acesso de grupos, entre outros dados importantes, é utilizado o banco de dados ArangoDB [144], um banco de dados híbrido, capaz de armazenar modelos relacionais e não relacionais simultaneamente dependendo da necessidade;
- **Orquestração (*middleware*):** coordena a comunicação da camada superior de interface de usuário com as camadas de mais baixo nível. É responsável por receber as requisições feitas pelos usuários através da GUI e repassá-las para os componentes que realizarão os comandos presentes nas requisições. A arquitetura do orquestrador é composta por um sistema de mensageria (NATS), com a função de receber as requisições de usuários e disponibilizá-las em um determinado tópico, no qual, um ou mais nodos de trabalho (*workers*) poderão acessar estes tópicos e executar as instruções presentes nas requisições utilizando as APIs nativas dos hipervisores, após isso, são gerados *logs* de trabalho que são armazenados em um repositório de metadados;
- **Gerenciamento de Recursos:** o gerenciamento de recursos é definido pelo administrador do sistema que pode determinar os limites de recursos disponíveis para cada grupo de acesso. Fica a cargo de cada grupo realizar o gerenciamento interno dos recursos disponibilizados. O gerenciamento de rede é automatizado pela ferramenta phpIPAM [145], integrando IPs e VLANs instanciadas pelo provedor local. Cada hipervisor fica responsável por realizar o gerenciamento dos volumes de bloco e o catálogo de imagens de VMs. Além disso, os recursos em nuvem oferecidos pelo provedor local podem ser provisionados diretamente pela plataforma Cloud.Jus;
- **Monitoramento:** o monitoramento de alocação de recursos em nuvem e saúde de ambiente permite que os administradores tenham uma visão geral do ambiente de nuvem, e também oferece aos usuários acesso à informações importantes sobre os recursos alocados e recursos utilizados, facilitando a organização e controle de uso de recursos internos. Cada VM possui um conjunto de métricas nativas que podem ser monitoradas por ferramentas como o Zabbix [146]. Para facilitar a exibição de dados de monitoramento, é utilizada o Grafana [147], uma ferramenta de exibição de métricas, capaz de consultar e visualizar métricas, além de configurar alertas com base em gatilhos customizáveis. O Grafana foi escolhido por sua fácil configuração e integração com serviços de monitoramento como o Zabbix, além de fornecer painéis e *dashboards* de monitoramento de maneira clara e de fácil entendimento, facilitando o

trabalho de monitoramento de ambiente por parte dos administradores do ambiente de nuvem;

- **Gerenciamento de Eventos:** o gerenciamento de eventos é feito pela ferramenta de monitoramento Zabbix, capaz de fazer a coleta de métricas de máquinas virtuais, servidores físicos, dispositivos de rede, além de diversos outras aplicações. Utilizando o Zabbix, é possível detectar incidentes e problemas diversos, utilizando gatilhos e alertas que podem ser configurados juntamente com ferramentas de visualização de métricas, como o Grafana, viabilizando o cumprimento dos SLAs estabelecidos.

5.4 Painel de Controle - *Dashboard* (GUI)

A plataforma de gerenciamento de nuvem Cloud.Jus oferece uma interface gráfica intuitiva e de fácil utilização. Visto que o objetivo da plataforma não é apenas ser utilizada por administradores do ambiente de nuvem, a interface gráfica do Cloud.Jus foi projetada para ser amigável ao usuário comum, de forma que um usuário sem conhecimento dos mecanismos internos de nuvem seja capaz de requisitar o provisionamento de recursos em nuvem sem que seja necessário auxílio de um especialista.

Assim sendo, por meio da interface de usuário, os usuários e administradores da plataforma podem realizar o provisionamento de recursos em nuvem, como máquinas virtuais, além de realizar a manipulação dos recursos já alocados em nuvem, como por exemplo, realizar a alteração de quantidade de memória virtual ou número de núcleos virtuais de uma determinada VM. Além disso, o usuário da plataforma também tem acesso à diversas informações sobre os recursos alocados em nuvem, como informações detalhadas sobre instâncias de VMs, ademais de uma visão geral dos recursos totais alocados e utilizados pelo grupo de acesso do usuário.

Para realizar o desenvolvimento da inteface de usuário, foi utilizada a linguagem de programação PHP [139], por sua simplicidade e agilidade, pode ser facilmente utilizada em conjunto com HTML5 [140]. Além disso, a linguagem PHP inclui um vasto conjunto de bibliotecas padronizadas que auxiliaram na integração com o ambiente de identificação e autorização LDAP [148] (gerenciamento de identidades) já utilizado no ambiente do Supremo Tribunal Federal (STF), local onde foi implantado a primeira versão do Cloud.Jus. Adicionalmente, foram feitos uso de *frameworks* como o Bootstrap [149], utilizando principalmente o *template* Gentelella [150], com o intuito de desenvolver uma interface *web* rica em detalhes, confortável e intuitiva para o usuário, como mostra a Figura 5.2.

A fim de facilitar o gerenciamento do ambiente de nuvem, cada *tenant* possui um ambiente único reservado no painel, além de possuir uma *pool* individual de recursos. Visto isso, cada ambiente no Cloud.Jus possui uma *dashboard* específica para cada *tenant*, de

forma que cada ambiente é totalmente isolado dos demais. Desta forma, os administradores podem administrar cada ambiente individualmente, gerenciando recursos em nuvem, VMs e acessos específicos de cada *tentant*. Para ilustrar a diferença entre ambientes, é possível observar que a *pool* de recursos em nuvem do *tenant* "vCloud", mostrado na Figura 5.2 é diferente e completamente isolado da *pool* de recursos em nuvem do *tenant* "VMware", mostardo na Figura 5.3.

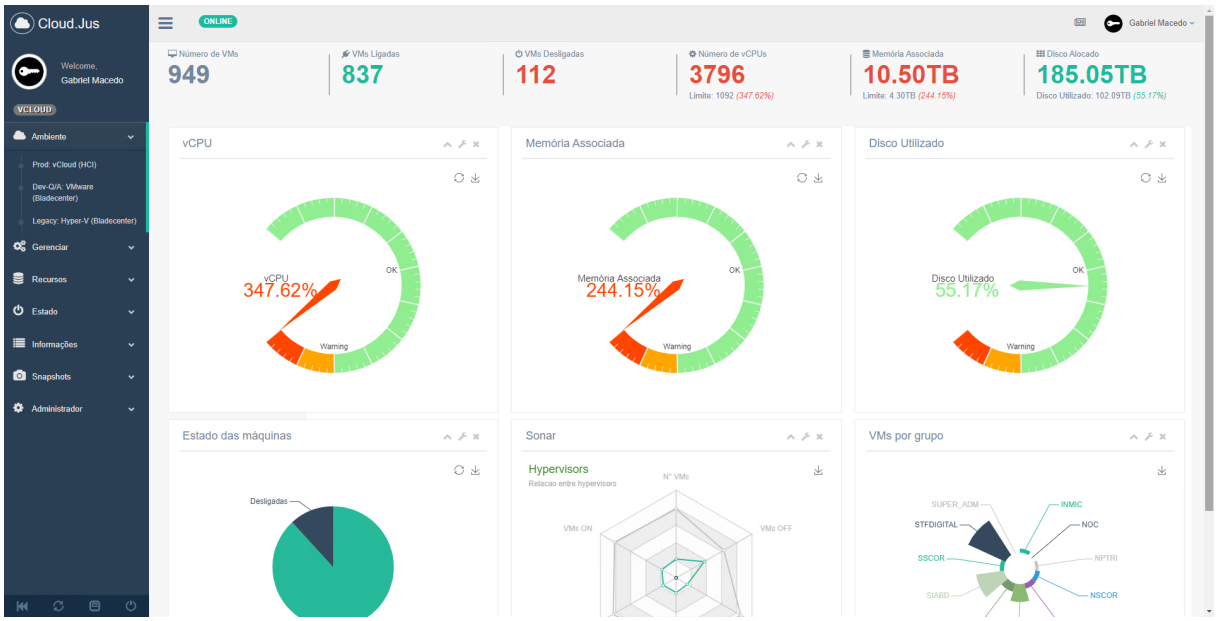

Figura 5.2: *Dashboard* do *tenant* "vCloud"na plataforma Cloud.Jus.

Através da *Dashboard* os usuários podem realizar requisições de criação de máquinas virtuais, alterações dos recursos virtuais disponíveis para máquinas específicas e exclusão de VMs. O Cloud.Jus disponibiliza diferentes modelos de VMs para atender diferentes necessidades. Também é possível realizar a requisição de uma VM customizada que diverge dos padrões pré-estabelecidos, no entanto, tal pedido deve ser aprovado por administradores do sistema.

A Figura 5.4 mostra a criação de uma máquina virtual no ambiente ou *tenant* "VMware". Observa-se que a VM será criada utilizando o *template* de imagem CentOS 7.7, no modelo "Medium", ou seja, a máquina será criada com uma vCPU, 6GB de memória RAM e 60GB de HD. Observa-se também que é possível determinar em qual *datacenter* a instância será criada e executada, no caso, a VM mostrada será criada no "*Datacenter* 01"do STF. O usuário também pode requisitar VMs em outros padrões de tamanho (*"Tiny, Small, Medium, Large* e *XLarge*), mas qualquer instância requisitada, que fuja dos padrões pré-estabelecidos, deve ser aprovada por administradores do sistema antes da execução da requisição.

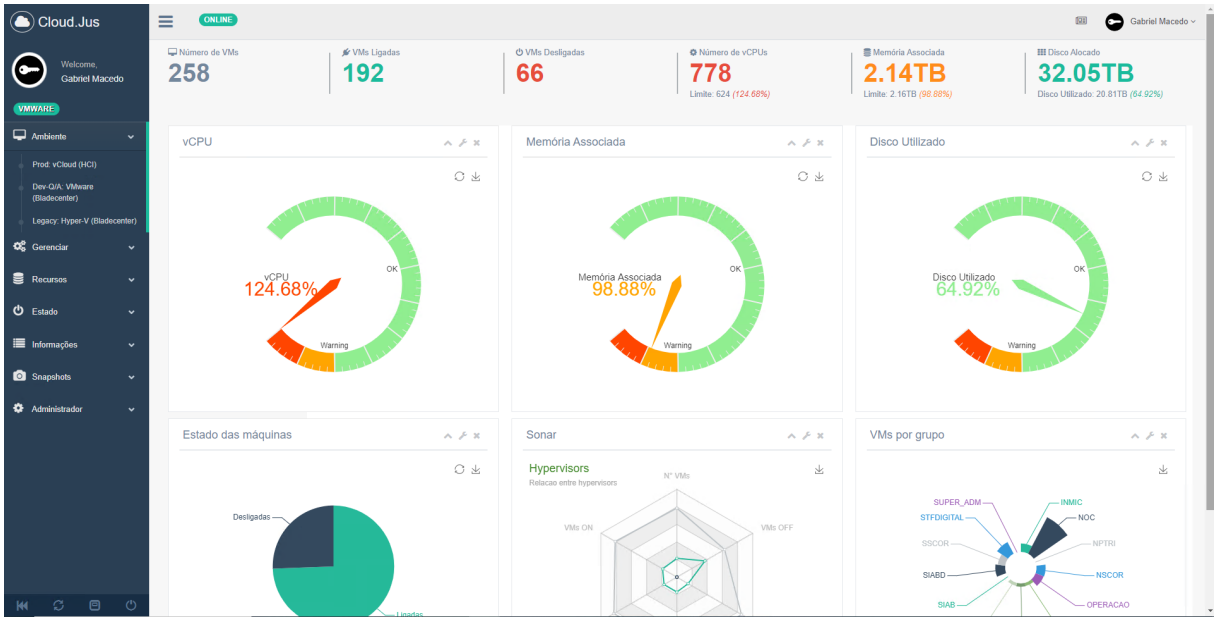

Figura 5.3: *Dashboard* do *tenant* "VMware"na plataforma Cloud.Jus.

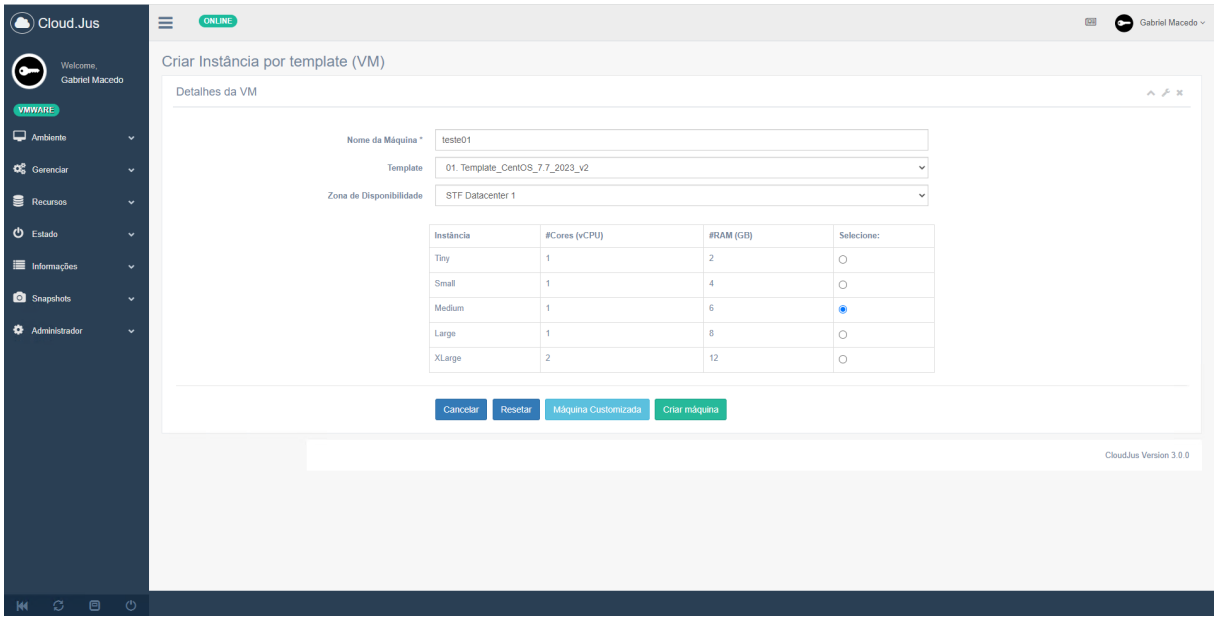

Figura 5.4: Criação de máquina virtual pelo *Dashbaord* da plataforma Cloud.Jus.

A plataforma também permite que os usuários realizem alterações nas máquinas virtuais existentes, como remoção da máquina virtual, alteração no número de vCPUs, alteração na quantidade de memória RAM ou acrescentar armazenamento de bloco. Alterações de recursos de VMs podem ser feitas acessando a aba de "Recursos"do *Dashbaord*, enquanto alterações de estado de instâncias, como ligar ou desligar uma VM, podem ser

feitas acessando a aba de "Estado". Na Figura 5.5 é mostrado um exemplo de como se alterar o número de vCPUs de uma determinada máquina virtual.

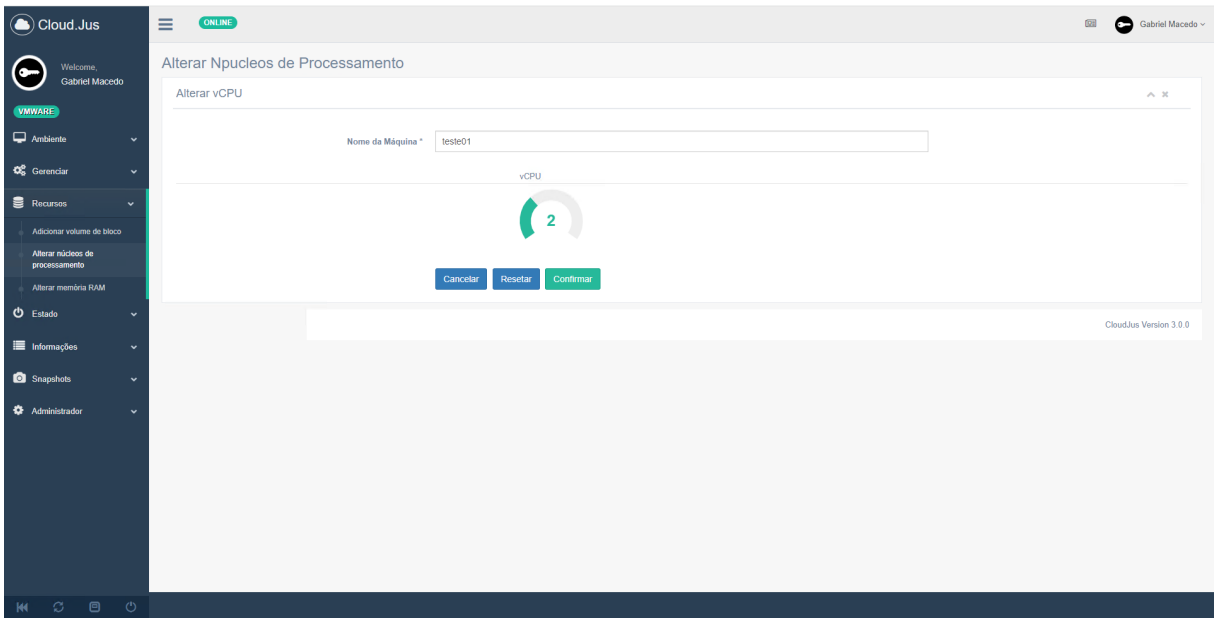

Figura 5.5: Ateração do número de vCPUs de uma determinada VM através do *Dashbaord* da plataforma Cloud.Jus.

A autenticação e autorização de usuários é integrada ao provedor de gerenciamento de identidades do ambiente, desta forma, cada usuário logado ao ambiente é atribuído à um grupo de controle de acesso. O grupo dos administradores de nuvem tem o nível máximo de acesso aos recursos de rede e às informações de alocação de recursos do ambiente como um todo. Além disso, os administradores de nuvem tem o poder de controlar as cotas de recursos que cada um dos grupos de controle de acesso possuem em cada ambiente, além de atribuir também níveis de acesso a cada grupo. As cotas e níveis de acesso de cada grupo são definidas individualmente por ambiente, ou seja, as cotas e níveis de acesso de usuários podem diferir entre *tenants*. Fica também à cargo dos administradores analisar pedidos de criação de máquinas virtuais customizadas requisitadas pelos usuários da plataforma, como mostra a Figura 5.6.

Cada grupo de acesso possui seu próprio *pool* de recursos e lista da de VMs por ambiente, como mostra a Figura 5.7. O provisionamento de recursos de forma automática por parte dos usuários dependerá da disponibilidade da cota estabelecida para o grupo em um determinado *tenant*. As cotas são definidas pelos administradores da plataforma e podem ser cotas do tipo *hard*, o qual obriga que os recursos requisitados estejam estritamente disponíveis dentro da cota estabelecida, ou podem ser cotas do tipo *soft*, ou seja,

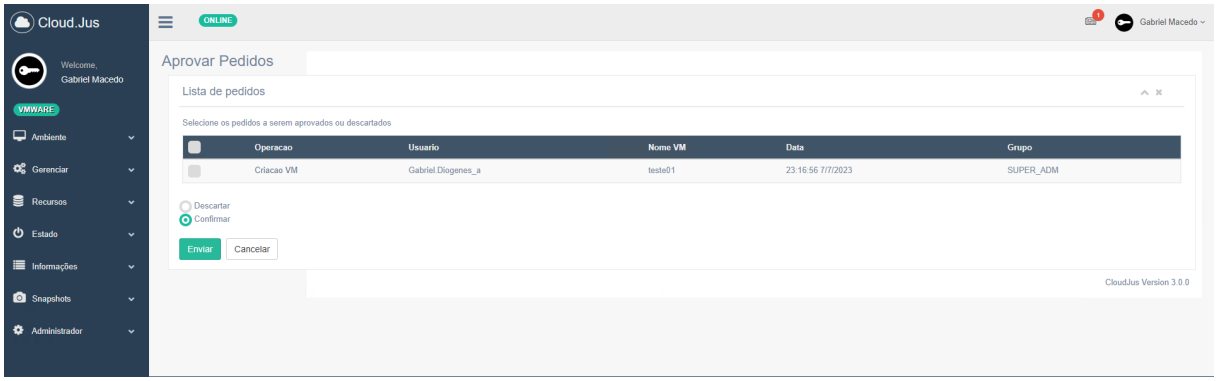

Figura 5.6: *Workflow* de aprovação de máquinas virtuais customizadas na plataforma Cloud.Jus.

as cotas são flexíveis e podem ser extrapoladas mediante um contrato prévio. Todas as cotas podem ser reajustadas de acordo com a necessidade da organização.

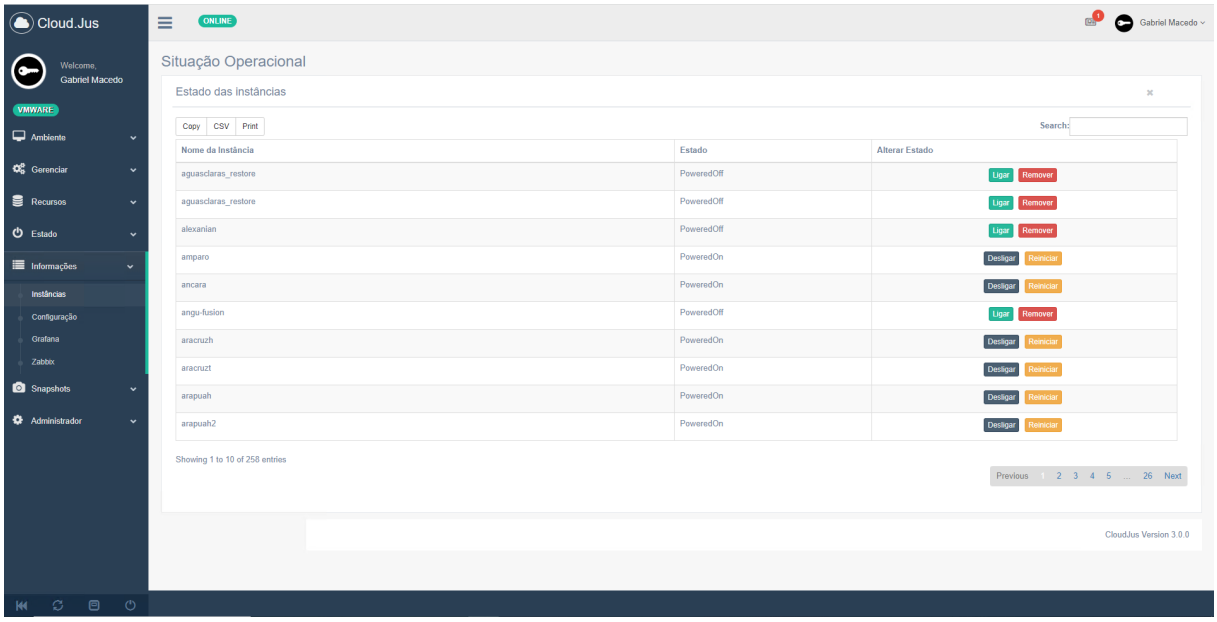

Figura 5.7: Lista de máquinas virtuais pertencentes de um determinado grupo no *tenant* "VMware".

Além disso, cada grupo recebe um nível de acesso para cada *tenant*. O nível de acesso determina as ações que os usuários de um determinado grupo podem realizar dentro da plataforma. Usuários pertencentes a grupos com um baixo nível de acesso podem apenas visualizar informações sobre as máquinas virtuais de seu grupo, realizar a alteração de estado das instâncias (ligar e desligar VMs) e realizar cópias instantâneas e reversões de estado (*snapshots*). Enquanto isso, grupos com um nível de acesso médio podem, além de modificar os estados das instâncias, provisionar mais recursos computacionais para as ins-

tâncias existentes. Por fim, grupos de maior acesso são capazes de criar máquinas virtuais padronizadas sob-demanda, sem necessidade de aprovação de administradores de nuvem, ou requisitar o providenciamento de máquinas virtuais customizadas sob aprovação dos administradores de nuvem.

5.5 *Application Programming Interface* (API)

Servindo como backend para a interface gráfica, a plataforma Cloud.Jus conta com uma API responsável por realizar a comunicação entre *frontend* e as camadas de mais baixo nível da plataforma, além de ser responsável por organizar os dados relevantes que vão ser mostrados na *Dashboard*. A API foi desenvolvida utilizando a linguagem de programação Javascript e o *framework* NodeJS. A API também utiliza o banco de dados ArangoDB [144] para auxiliar no armazenamento e organização de dados relevantes que são necessários para o funcionamento da interface de usuário ou dados que vão ser mostrados no *Dashboard*.

5.5.1 A Linguagem Javascript

A linguagem de programação Javascript [141] é uma das linguagens mais utilizadas no mundo. Ela é conhecida por ser a linguagem da web, a ampla maioria dos *sites* a utiliza em algum nível do desenvolvimento, além de poder ser interpretada por virtualmente todos os navegadores modernos, computadores pessoais, *smartphones*, entre outros diversos aparelhos modernos, a tornando a linguagem de programação mais utilizada do mundo [141].

O Javascript é uma linguagem de alto nível, dinâmica, não-tipada, interpretada e que suporta o modelo de programação orientada a objetos [141]. O Javascript foi derivado da linguagem Java [151] e seu nome, "Javascript", apesar de conter o nome "Java"em sua composição, é extremamente diferente da linguagem Java. Adicionalmente, apesar da Javascript ter sido uma linguagem do tipo *script* durante suas primeiras versões, há muito tempo o Javascript deixou de ser uma linguagem de *scripts*, se tornando uma linguagem de uso geral.

Apesar de inicialmente o Javascript tenha sido criado com o intuito de ser utilizado em aplicações web, a linguagem evoluiu para ser uma ferramenta útil em diversas outras áreas da programação. Atualmente, o Javascript é padrão não só em desenvolvimento de aplicações web, visto que diversos *frameworks* utilizam a linguagem, sendo utilizada em *frameworks* de desenvolvimento *web, mobile* e até mesmo de aplicações de *backend*, como APIs REST. Um dos *frameworks* que utiliza a linguagem Javascript mais utilizados para desenvolvimento de APIs REST é o NodeJS [142], que será detalhado a seguir.

5.5.2 NodeJS

Com o crescimento da popularidade da linguagem Javascript, Rayan Dahl, propôs em 2009 a criação do *framework* NodeJS [152]. No entanto, o NodeJs se propõe a ser mais do que só um simples *framework*, mas não a ponto de ser considerado uma linguagem de programação nova, sendo definida como um ambiente execução de código Javascript [152]. A proposta do NodeJS é facilitar o desenvolvimento de aplicações altamente escaláveis, oferecendo suporte à execução de processos assíncronos, permitindo que outros processos continuem sua execução até mesmo quando a aplicação está aguardando a resposta de uma chamada [152].

O NodeJS conta com uma vasta comunidade de suporte que oferece uma grande quantidade de bibliotecas de código aberto, permitindo o reaproveitamento de código. Usuários do NodeJS podem acessar as bibliotecas disponibilizadas pela comunidade através do *Node Package Manager* (NPM) [153]. O NPM permite que seus usuários possam baixar e instalar diversas bibliotecas em NodeJS desenvolvidas por terceiros, permitindo utiliza-las em sua aplicação. O NPM permite a fácil publicação de bibliotecas em sua base de dados, criando uma cultura de reaproveitamento de código por toda a comunidade. Uma das bibliotecas utilizadas no desenvolvimento da API REST do Cloud.Jus foi a biblioteca *express* [154].

5.5.3 ArangoBD

O ArangoDB [155] é uma sistema de gerenciamento de banco de dados para armazenamento de grafos, também contendo diversas funcionalidades e um rico ecossistema. No entanto, o ArangoDB também suporta diversos outros modelos de dados, como dados estruturados, semi-estruturados e não estruturados que podem ser interpretados a partir de em formato JSON [156]. O ArangoDB foi desenvolvido na linguagem C++ [157] e projetado para alta performance, escalabilidade em ambientes locais ou em nuvem.

O ArangoDB foi escolhido como o banco de dados da plataforma Cloud.Jus por ser um banco de dados híbrido capaz de se adaptar à diferentes modelos de dados [155]. Por exemplo, o ArangoDB suporta modelos de dados do tipo grafo, modelos relacionais, tipo documento, tipo chave-valor e é possível até mesmo utiliza-lo como motor de busca ou para aprendizado de máquina. Visto que, a plataforma Cloud.Jus necessita de um banco de dados capaz de se adaptar aos ambientes heterogêneos de nuvem onde a plataforma é utilizada, por esse motivo, o ArangoDB foi escolhido.

Dessa forma, foi possível tomar proveito do suporte aos diversos modelos de dados suportados pelo ArangoDB, visto que diferentes componentes da plataforma utilizam diferentes modelos de dados. Por exemplo, são armazenados no ArangoDB as listas de

máquinas virtuais no modelo de documento, enquanto os dados de controle de acesso dos usuários é armazenado de forma relacional. Adicionalmente, até mesmo os processos das camadas de mais baixo nível utilizam o recurso de armazenamento de dados do tipo chave-valor que ArangoDB oferece, a fim de auxiliar no controle de execução de requisições de provisionamento de recursos em nuvem.

5.5.4 Arquitetura da API do Cloud.Jus

A API do Cloud.Jus foi construída para dar suporte de *backend* para a interface web da plataforma. Através da API, o *Dashboard* pode realizar a requisição de dados importantes, utilizando o formato JSON [156], como listas de máquinas virtuais, que vão ser mostrados em tela. Para realizar o armazenamento de dados, a API se comunica com o ArangoDB que contém todos os dados necessários para o funcionamento do *Dashboard*.

Além disso, a API também é responsável para realizar o redirecionamento de requisições de provisionamento de recursos em nuvem feitos pelos usuários da plataforma. Ao receber uma requisição de provisionamento, também em formato JSON, a API realiza a publicação da requisição em um tópico do serviço de mensageria que esta sendo executado na camada de abstração, a partir daí fica a cargo das camadas de mais baixo nível processarem a requisição.

Adicionalmente, fica a cargo da API receber os dados das camadas mais baixo nível da plataforma a fim de popular e atualizar os dados presentes no ArangoDB. As camadas de baixo nível possuem processos responsáveis por executar as requisições de provisionamento e acessar os dados e informações sobre as instâncias que estão sendo executadas nos ambientes de nuvem. Tais processos podem acessar e enviar os dados e informações sobre as VMs para a API que por sua vez armazena estes dados no banco de dados. Desta forma, caso seja necessário mostrar tais dados no *dashboard*, é necessário apenas uma consulta no banco de dados, sem necessidade de acessar as camadas inferiores para buscar tais dados. As rotas disponibilizadas pela API são divididas em quatro categorias:

- **Recursos:** nesta categoria estão contidas as rotas responsáveis por organizar e retornar informações sobre os recursos disponíveis em nuvem, tais como listas de *datacenters* e *templates* de máquinas virtuais disponíveis;
- **Máquinas Virtuais:** nesta categoria estão contidas as rotas responsáveis por receber informações sobre as máquinas virtuais vindas das camadas de baixo nível da plataforma e retornar estas informações para o *Dashboard*;
- **Grupos:** esta categoria de rotas tem a função de retornar ao *Dashboard* informações importantes sobre as regras de acesso de grupo de usuários e cotas de recursos disponíveis para cada grupo dentro de um determinado *tenant*;

- **Requisições:** é o conjunto de rotas responsáveis por receber as requisições de provisionamento de recursos em nuvem, e redirecioná-las ao sistema de mensageria para que a requisição seja processada pelas camadas de baixo nível da plataforma.

A API integrada com o banco de dados ArangoDB foi uma das melhorias implementadas na versão 3.0 do Cloud.Jus. Em versões anteriores, muitos dados eram armazenados em formato de texto em sistema de arquivos (FS - *File System*), que apesar de ser de acesso fácil e rápido, não era uma solução escalável. Ao integrar a API e o banco de dados ao *Dashboard*, a plataforma se tornou mais modular e desacoplada, de forma que hoje é possível realizar a conteinerização dos componentes da plataforma e implantá-los em outro ambiente. Adicionalmente, com o uso do banco de dados, a atualização dos dados das instâncias se tornou mais rápido e fácil, visto que, cada máquina virtual possui um documento armazenado no banco de dados contendo seus dados operacionais, ao contrário de como acontecia em versões anteriores, em que os dados de todas as máquinas virtuais eram armazenados em um único documento de texto, o que tornava a atualização destes dados extremamente custosa e lenta.

5.6 Orquestrador - *Middleware*

A plataforma Cloud.Jus oferece aos seus usuários o provisionamento de máquinas virtuais padronizadas em um determinado *tenant*, de forma automática caso o usuário tenha as permissões necessárias. No entanto, o processo de provisionamento de recursos em nuvem permanece o mesmo, ou seja, a função da plataforma é a automatização e a padronização da interação entre os componentes, de forma que, todo o processo ocorra de forma transparente para o usuário.

Visto isso, foi projetado um componente de *middleware* com a função de realizar a comunicação entre as camadas mais alto nível (GUI e API) com as camadas de mais baixo nível (hipervisores) da plataforma. Desta forma, o *middleware* é responsável pelo gerenciamento automatizado das tarefas e procedimentos necessários para o funcionamento de ambiente, desde a coordenação, organização e transporte das requisições vindas do *front-end* até a organização e armazenamento de logs de atividade de provisionamento.

O *middleware* inicialmente foi projetado utilizando apenas *scripts* Powershell [138] que realizava o gerenciamento de logs de ações de provisionamento e o balanceamento de carga das requisições, encaminhando-as para outros módulos *workers*, como mostra a Figura 5.8. Cada módulo *worker* é um *script* Powershell com a função de receber requisições do *middleware* e comunica-las para o hipervisor de seu determinado *tenant*, para que cada requisição seja executada e os recursos sejam devidamente provisionados. Após o provisionamento do recurso ser realizado, o *worker* que executou a requisição

informa de volta para o *middleware* que a execução do pedido foi bem sucedida e o *middleware* atualiza o repositório de metadados com os dados da requisição bem sucedida e envia os dados atualizados sobre os recursos e status do ambiente (informações sobre VMs e recursos alocados) de volta para a API, para que os dados presentes no banco de dados estejam sempre atualizados, como mostra a Figura 5.8.

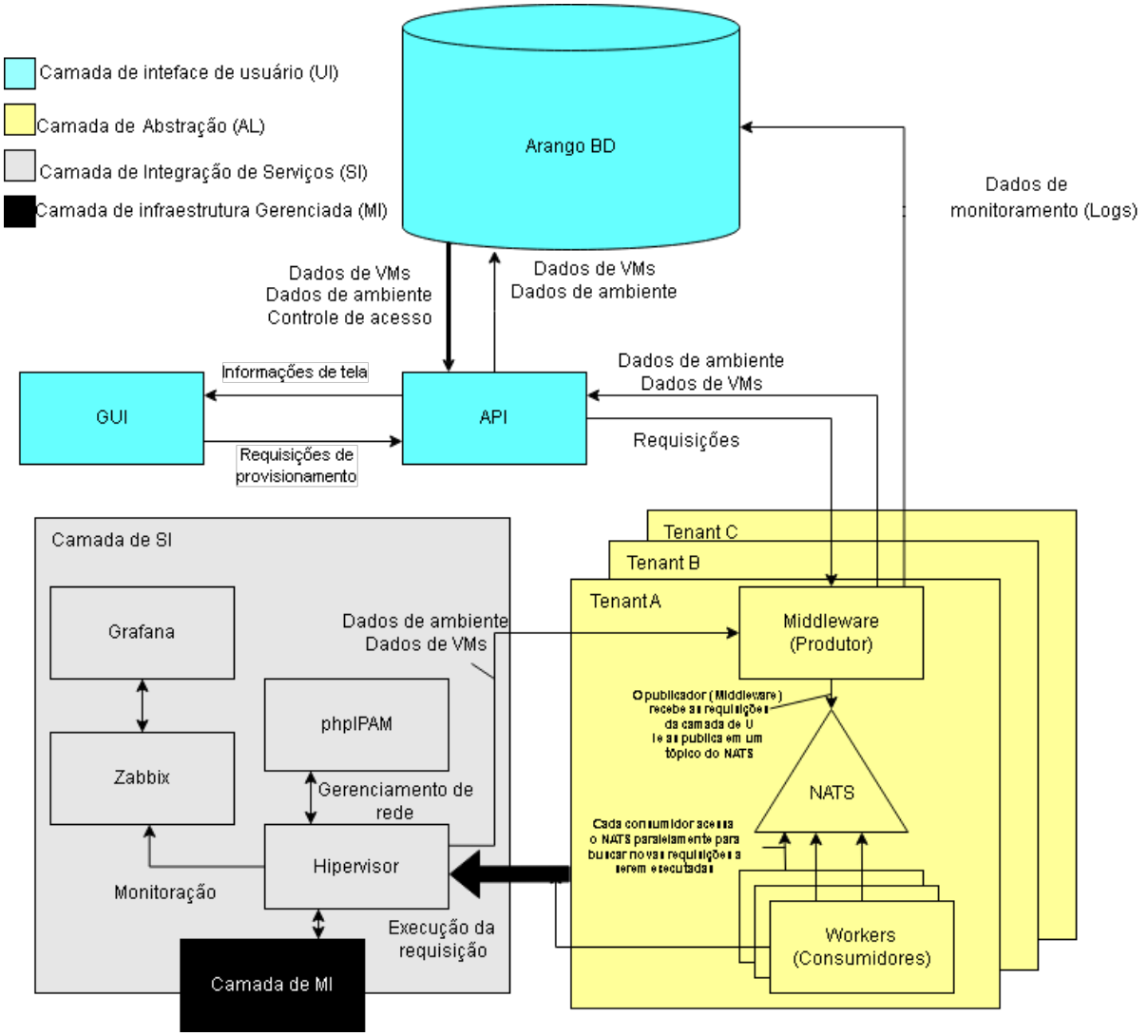

Figura 5.8: Arquitetura da Plataforma de Gerenciamento de Nuvem Cloud.Jus sem a utilização de serviço de mensageria.

Após a implementação da primeira versão do Cloud.Jus, observou-se que o componente *middleware* possuía uma carga de processamento elevado, pelo motivo de ter a função de realizar a orquestração e balanceamento de uma grande carga de requisições e realizar o gerenciamento do repositório de metadados simultaneamente. Tal fato, se tornou preocupante, pois percebeu-se que tal solução não era escalonável e poderia ser um possível

gargalo de eficiência na plataforma em ambientes de alta demanda. Visto isso, para a versão 3.0 do Cloud.Jus, apresentada neste trabalho, foi proposto uma nova abordagem para que seja realizada o balanceamento de carga de requisições. A solução proposta foi adicionar um serviço de mensageria entre o *middleware* e os módulos *workers*, desta forma, a carga de balanceamento de requisições cairia sobre um sistema de mensageria que possui mecanismos internos para a otimização na organização e balanceamento de grandes cargas de requisições.

Após um período de experimentação e pesquisa, foi escolhido o serviço de mensageria NATS [128] para realizar a orquestração e balanceamento de carga de requisições. Desta forma, o mecanismo de balanceamento estaria desacoplado do orquestrador *middleware*, paralelizando a execução de tarefas entre um numero maior de módulos *workers* e melhorando a escalabilidade do sistema.

Com a adoção do NATS na função principal de comunicação entre os componentes na camada de abstração da plataforma, a solução de fato se tornou inteiramente desacoplada e escalável, visto que, apesar da solução anterior buscasse balancear e paralelizar a carga de trabalho, o *middleware* ainda tinha a função de realizar o balanceamento e redirecionamento de cada requisição individualmente de forma serial, limitando a sua capacidade de expansão. Visto isso, ao adotar um sistema de mensageria distribuída, o problema de sobrecarga do *middleware* pôde ser resolvido, pois o NATS apresenta uma baixa sobrecarga em termos de protocolo.

Um dos motivos que o NATS diminui a sobrecarga de processamento ao realizar o gerenciamento e o balanceamento de requisições está na forma com que o NATS realiza o tráfego e o balanceamento de requisições entre os módulos *workers*. Na abordagem utilizada pelo NATS, não há necessidade do próprio serviço de mensageria estabelecer uma conexão direta ponto-a-ponto com cada um dos módulos *workers* para realizar a transferência de requisições. Pelo contrário, é função dos próprios módulos *workers* e *middleware* inciar a comunicação com o serviço de mensageria. O NATS organiza esta comunicação separando os módulos entre produtores e consumidores, de forma que o *middleware* faz o papel de produtor ao se conectar ao servidor do NATS para realizar a publicação das requisições vindas do *front-end*, enquanto os módulos *workers* realizam o papel de consumidores ao se comunicar com o servidor NATS para consumir estas requisições publicadas pelo *middleware*, como apresentado na Figura 5.9, descentralizando a carga de trabalho e aumentando a escalabilidade.

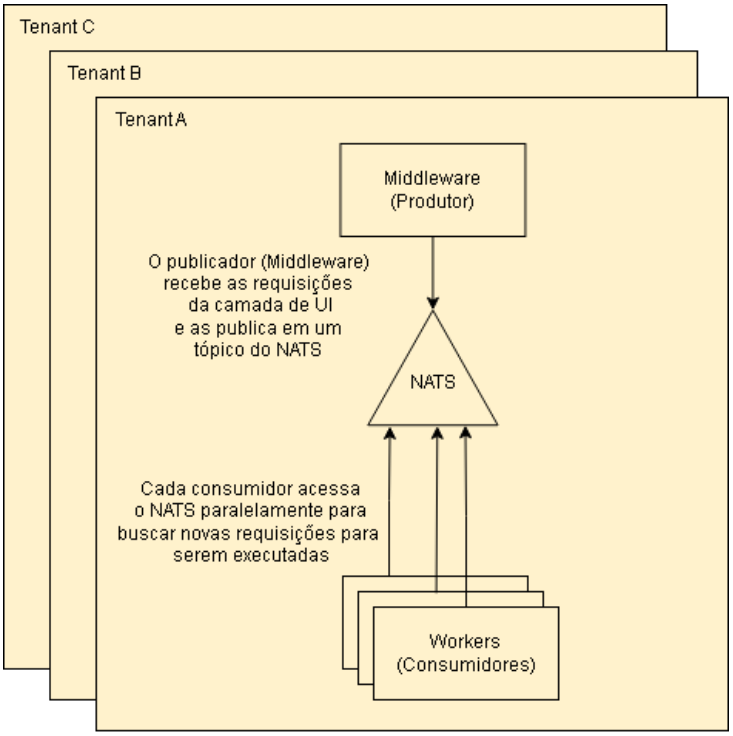

Figura 5.9: Camada de abstração da plataforma Cloud.Jus, após a implantação do serviço de mensageria NATS.

5.7 Gerenciamento de Recursos

Como foi mostrado na seção anterior, o componente *middleware* é o responsável por manter a base de metadados de requisições e a base de dados contendo as informações de ambiente atualizadas, para que o *Dashboard* possa acessar estes dados de forma assíncrona, através da API, de maneira rápida, sem necessidade de acesso à camadas de mais baixo nível da plataforma. Além disso, as requisições enviadas através da interface de usuário são organizadas pelo *middleware*, com o apoio de um sistema de mensageria, e distribuídas entre módulos de *back-end* (*workers*) para serem processadas.

Todo processo é feito de maneira inteiramente desacoplada ao funcionamento das camadas mais baixo nível da plataforma, como a Camada de Integração de Serviços e a Camada de Infraestrutura Gerenciada. A interface se comunica com os outros componentes do sistema utilizando requisições em formato JSON [156], sendo que cada requisição gerada pela GUI será processada por um módulo *worker* que se comunica com o hipervisor por meio de *scripts* Powershell. Cada requisição contem os dados necessários para a execução correta do provisionamento de recursos e em qual *tenant* a requisição será executada. Apesar do formato das requisições serem semelhantes entre si, os módulos *workers* tratam cada operação de provisionamento separadamente, variando conforme o

hipervisior utilizado.

Após a conclusão da execução da requisição, o módulo *worker* faz o envio de um e-mail contendo as credenciais para acessar o novo recurso provisionado através do endereço eletrônico do usuário contido na requisição. Após receber as credenciais do novo recurso, o usuário pode se conectar diretamente à nova VM instanciada via console, onde um novo IP e nome DNS podem ser atribuídos à ela. Assim então, o usuário pode realizar a inicialização da nova VM utilizando a imagem de sistema escolhida (Windows Server, Ubuntu, Oracle Linux ou CentOS) que realiza a execução de um *script* de autoconfiguração após o primeiro login realizado no sistema.

Para a gerência de rede foi utilizada a ferramenta de código aberto phpIPAM (*IP Address Management*) [145] para realizar automação de endereçamento de rede das máquinas virtuais. o phpIPAM realiza o endereçamento de VMs pré-reservando blocos de endereçamento IP nos segmentos de rede interna (DMZ - *Demilitarized Zone*) já com o seu respectivo NAT (*Network Address Translation*) e com liberação de acesso externo por meio dos protocolos SSH e HTTP/HTTPS para os usuários da rede comunitária. As máquinas virtuais que devem ser acessadas de maneira externa são hospedadas no segmento de rede DMZ, de forma isolada da rede interna do provedor. Para realização da comunicação entre a VM hospedada em rede privativa (DMZ) para a Internet, o *firewall* do provedor realiza a troca de endereço IP de origem (IP interna da VM) pelo endereço IP real e depois encaminha a transmissão.

5.8 Monitoramento

O componente de monitoramento foi projetado para que usuários e administradores pudessem realizar o acompanhamento da quantidade de recursos disponíveis e saúde geral do sistema [137]. Para que os usuários e administradores pudessem ter acesso fácil aos dados de monitoramento, a plataforma Cloud.Jus foi integrada com as ferramentas de monitoramento Zabbix [146] e Grafana [147] via API. A escolha destas ferramentas se deu por serem ferramentas robustas e flexíveis, além de serem facilmente integradas com outras ferramentas e componentes da plataforma Cloud.Jus por meio de RESTful APIs. Além disso, os administradores do sistema já possuíam conhecimento destas ferramentas de monitoramento por terem sido implantadas há um certo tempo no ambiente, facilitando a operação das ferramentas e estabelecimento de novas métricas de monitoramento.

A ferramenta Zabbix tem a função de monitorar o desempenho e a disponibilidade dos serviços e ativos em nuvem sem muito esforço na configuração de métricas. o Zabbix é capaz de realizar a coleta de métricas utilizando diferentes meios, como descoberta automática, *scripts*, SNMP (*Simple Network Management Protocol*) e IPMI (*Inteligent*

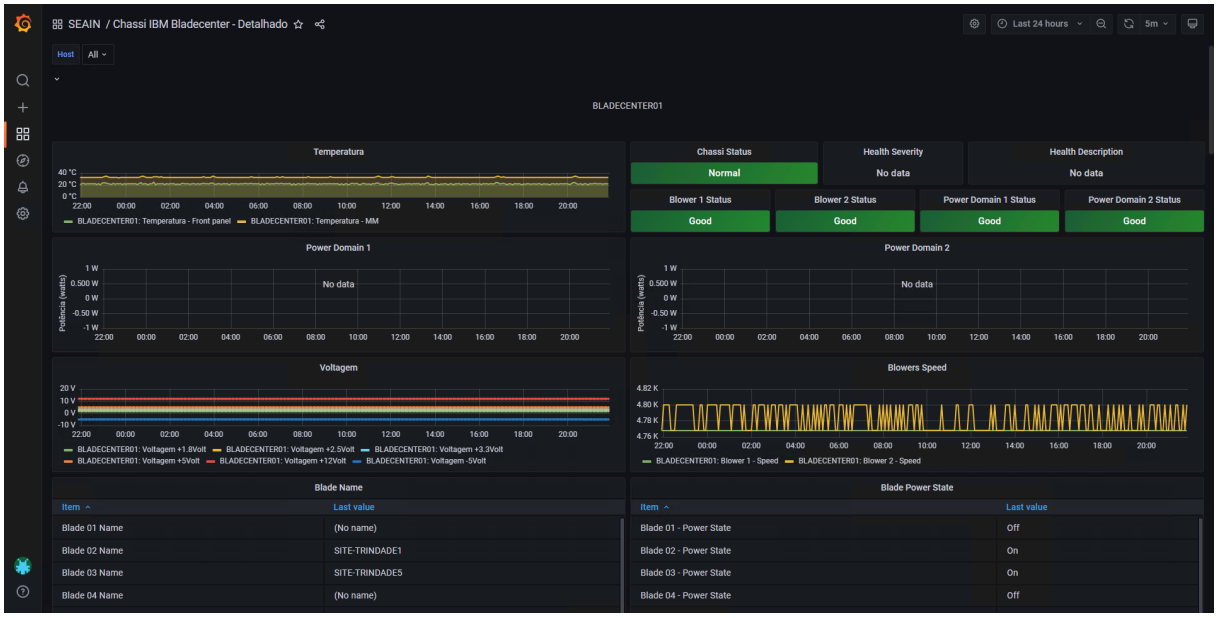

Figura 5.10: *Dashboard* Grafana mostrando métricas e dados coletados pelo Zabbix.

Plataform Management Interface). Assim, por meio destas fontes, o administrador do sistema pode utilizar o Zabbix para realizar a coleta de um conjunto extenso de métricas.

Trabalhando de forma integrada com o Zabbix, a ferramenta Grafana tem a função de apresentar os dados coletados em forma de painéis configurados utilizando expressões regulares, como mostra a Figura 5.10. Os dados coletados pelo Zabbix são refletidos nos *dashboards* do Grafana de forma imediata, por meio da integração nativa entre as duas ferramentas, realizada por meio de API. Desta forma, administradores de nuvem podem realizar a monitoramento da saúde do ambiente em tempo real, tendo acesso a dados como utilização de CPU, uso de memória, consumo de disco, tráfego de rede, entre diversas outras métricas.

5.9 Gerenciamento de Eventos

O componente de Gerenciamento de Eventos foi criado para realizar a detecção de incidentes de infraestrutura e ações de contingenciamento, a fim de viabilizar o cumprimento dos SLAs estabelecidos [137]. Como mostrado da Sessão anterior, a ferramenta de monitoramento Zabbix é responsável pelo monitoramento do ambiente como um todo. Após a coleta e persistência dos dados em forma de séries temporais (dados coletados e organizados sequencialmente ao longo do tempo), os dados são organizados de acordo com as associações a seguir:

- Criação de *hosts*, ou seja, ativos a serem monitorados;

- Criação de itens de monitoramento, ou seja, criação de novas métricas de coleta de dados;
- Criação de *triggers*, ou seja, criação de regras, ou gatilhos, que analisam uma certa métrica e ao atingir um certo valor, um evento é gerado. Os níveis de evento podem ser categorizados como "Informação", "Atenção", "Médio", "Alto"e "Desastre";
- Criação de ações, ou seja, caso algum gatilho seja disparado, o sistema realiza uma ação, como disparar notificações ou realizar algum comando remoto.

5.10 Avaliação experimental

Um dos pilares mais importantes da computação em nuvem é a elasticidade, ou seja, a capacidade de um sistema alterar o número de instâncias ou VMs dentro de um determinado ambiente de acordo com a necessidade dos usuários [24]. Visto isso, o foco da avaliação experimental será testar a capacidade da solução proposta em se adaptar à criação de dezenas de máquinas virtuais em rajada, a fim de verificar se a plataforma é capaz de realizar o provisionamento de uma grande quantidade de recursos de forma rápida e eficiente.

Testes semelhantes a este foram realizados na versão 2.11.6 da plataforma Cloud.Jus [27], no entanto, esta versão do Cloud.Jus não recebeu certas melhorias apresentadas neste capítulo, tais como os componentes de API, banco de dados e serviço de mensageria. Visto isso, o objetivo principal desta avaliação experimental é verificar se houve ganho ou perda de eficiência em termos de elasticidade entre as versões 2.11.6 e 3.0.0 do Cloud.Jus, apresentado neste trabalho.

5.10.1 Comparação entre Versões da Plataforma Cloud.Jus

O objetivo principal deste trabalho foi apresentar melhorias para a plataforma Cloud.Jus a fim de tornar seus componentes mais desacoplados e modulares, além de melhorar a eficiência da plataforma como um todo. A principal diferença entre a versão anterior com a versão 3.0.0 atual é a forma de armazenamento e transporte dos dados e requisições entre componentes.

Primeiramente, a versão 2.11.6 do Cloud.Jus não possuía uma API e banco de dados para realizar a o transporte e armazenamento de dados entre a Camada de Interface de Usuário e a Camada de Abstração. Toda a comunicação entre o componente GUI e o componente *middleware* era realizado utilizando arquivos texto em formato JSON [156] e a transferência de arquivos entre componentes era feita através de sistema de arquivos (FS - *File System*) de rede. Apesar de ser uma solução simples, funcional e com um rápido

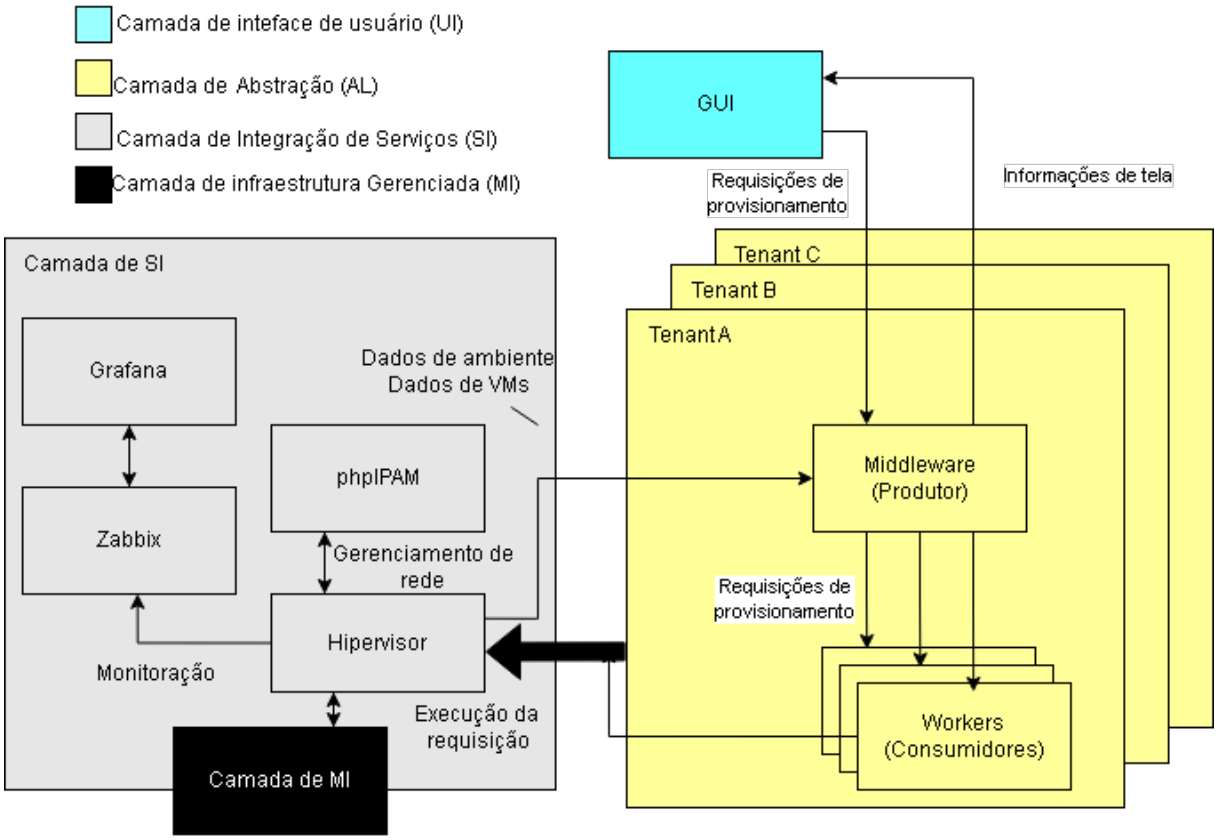

Figura 5.11: Arquitetura da versão 2.11.6 da Plataforma Cloud.Jus.

tempo de resposta para o cliente, visto que os dados a serem mostrados no *dashboard* já estão em disco, a solução não é escalonável, é altamente acoplada e pouco modular.

Adicionalmente, a versão 2.11.6 do Cloud.Jus não possui o serviço de mensageria para auxiliar o componente *middleware* a distribuir as requisições entre os módulos *workers* da Camada de Abstração, forçando o componente a realizar todo o balanceamento de carga, transferindo cada arquivo de requisição individualmente para cada módulo *worker*, causando picos de processamento e se tornando um possível gargalo no sistema. Além disso, os arquivos contendo os dados de ambiente, tal como listas de VMs, eram completamente centralizados, não permitindo a atualização de dados individuais das VMs. Com isso, a única forma de atualizar dados como listas de VMs instanciadas no ambiente era realizar a atualização completa da lista, causando retrabalho e tornando os processos de criação e remoção de VMs mais lentos. Tal limitação também causava uma grande assincronia entre os dados armazenados e mostrados na GUI e os dados reais do ambiente, visto que a atualização de dados de ambiente são altamente custosos. Visto isso, a atualização as listas de dados de ambiente eram feitas em determinados horários ou sob demanda dos administradores de nuvem, a fim de evitar a sobrecarga do *middleware*. A arquitetura da

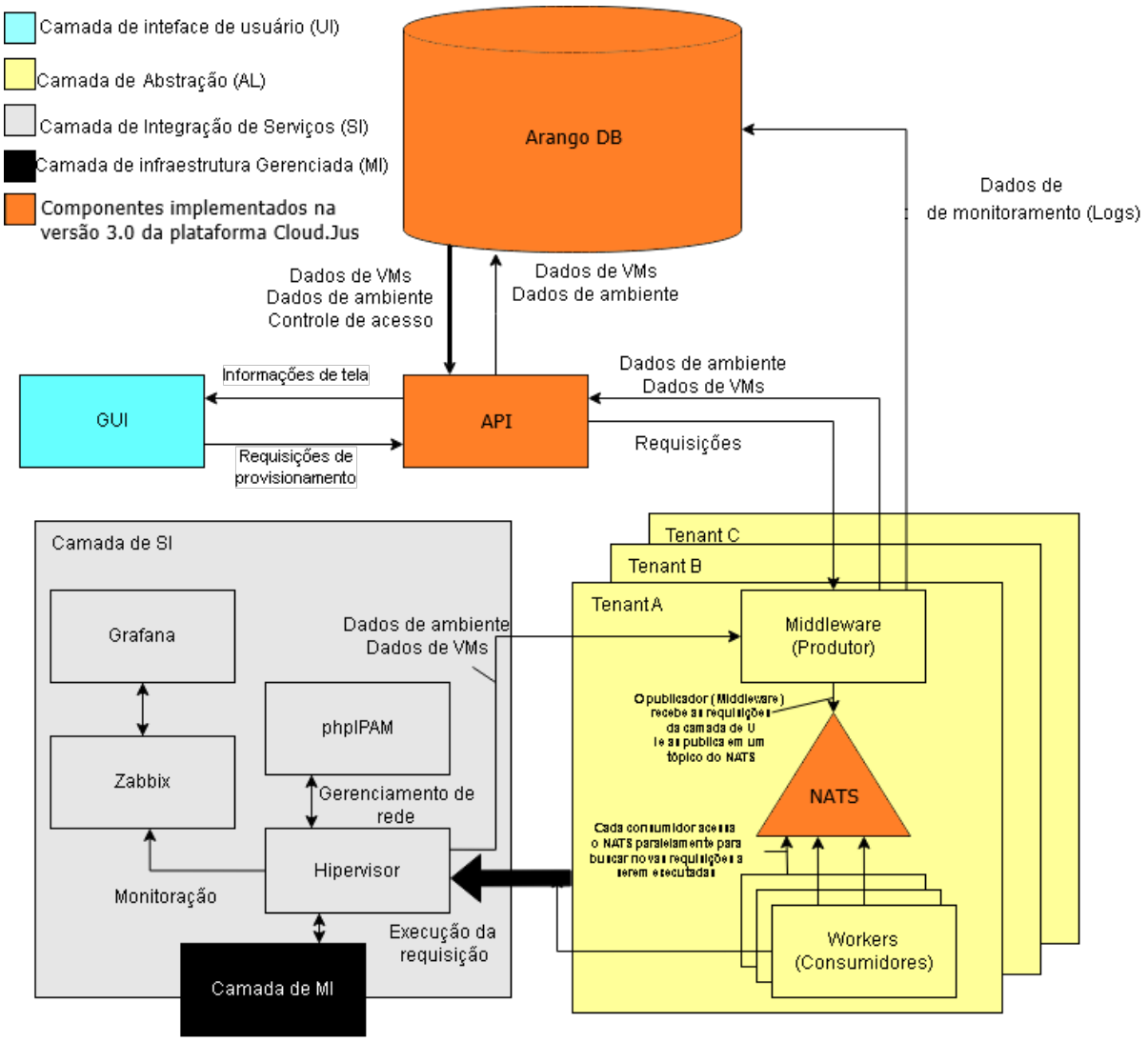

Figura 5.12: Arquitetura da Plataforma de Gerenciamento de Nuvem Cloud.Jus3.0.

versão 2.11.6 do Cloud.Jus pode ser observada na Figura 5.11.

A versão 3.0 do Cloud.Jus, como descrita nas Sessões anteriores, recebeu melhorias, como uma API e banco de dados para auxiliar no controle e transporte de requisições entre a GUI e a Camada de Abstração, além de facilitar a atualização dos dados de ambiente. Tomando como exemplo, as informações de VMs agora podem ser atualizadas individualmente, sem necessidade de realizar a atualização completa das listas de informações das VMs. Adicionalmente, com a implantação do serviço de mensageria, não há mais sobrecarga no componente de *middleware*, além de simplificar a organização e armazenamento de logs. A arquitetura completa do Cloud.Jus3.0 pode ser observada na Figura 5.12

5.10.2 Cenários de Teste

A fim de avaliar a performance das duas versões da plataforma, foram conduzidos 3 (três) cenários de teste de estresse, a fim de simular picos de utilização da plataforma, como, por exemplo, a requisição de provisionamento de dezenas de máquinas virtuais em lote de forma simultânea. Para todos os testes foram provisionadas instância no modelo padrão da plataforma Cloud.Jus, utilizando o *template* do sistema operacional CentOS 7.7.

Durante os experimentos foram utilizados 3 (três) módulos *workers* rodando *scripts* referentes ao *tenant* "vCloud". Cada *script* executa nodos do VMware ESXi 6.5 [158] com a função de despachar as requisições para o hipervisior VMware vCenter que realizará o provisionamento das máquinas virtuais. Cada *script* será executado em uma máquina virtual *worker* com 128G de armazenamento, 4 vCPUs Intel Xeon 2.4Gz e 8Gb de memória RAM cada. Os testes foram executados nas versões 2.11.6 e 3.0.0 da plataforma Cloud.Jus e foi monitorado o tempo de execução das requisições e o tempo de atualização do banco de dados para conter os dados das novas VMs provisionadas utilizando *scripts* Powershell.

- **Experimento 1:** 20 requisições de provisionamento de VM, um processo rodando por módulo *worker* referente ao *tenant* "vCloud"e apenas três módulos *workers* sendo executados;
- **Experimento 2:** 40 requisições de provisionamento de VM, um processo rodando por módulo *worker* referente ao *tenant* "vCloud"e três módulos *workers* sendo executados;
- **Experimento 3:** 60 requisições de provisionamento de VM, um processo rodando por módulo *worker* referente ao *tenant* "vCloud"e três módulos *workers* sendo executados.

5.10.3 Resultados e Discussão

Os três experimentos foram realizados e os resultados podem ser observados na Tabela 5.1. Durante a realização dos testes, cada componente da Camada de Integração foi monitorado. Observou-se que a média de utilização de CPU por parte dos módulos *workers* oscilou entre 6% e 10% durante períodos ociosos, geralmente, aguardando resposta do hipervisor, e de 16% a 36% em momentos de alto processamento, chegando em picos de até 53% de utilização de CPU. Na versão 2.11.6, em períodos de alto processamento, como por exemplo, na atualização da base de dados de métricas de ambiente, o componente *middleware* chegou a utilizar entre 6% e 10% de uso da CPU, enquanto na versão 3.0.0, o *middleware* permaneceu ocioso, mantendo-se entre 1% e 2% de utilização

Tabela 5.1: Resultado dos experimentos realizados

Experimento	**Tenant**	**Versão da Plataforma**	**Tempo de Execução (Worker)**	**Tempo de Atualização de Dados (Middleware)**	**Throughput (VMs/min)**	**Tempo total**	**Sumário**
Experimento 1	vCloud	2.11.6	6min18seg	18min	3,17	24min18seg	20 requisições, 1 worker, 3 processos, 4vCPUs, sistema de arquivos, atualização de dados centralizada
		3.0.0	7min48seg	–	2,56	7min48seg	20 Requisições, 3 workers, 3 processos,4vCPUs, serviço de mensageria, atualização de dados descentralizada
Experimento 2	vCloud	2.11.6	12min6seg	18min17seg	3,30	30min23seg	40 Requisições, 3 workers, 3 processos,4vCPUs, sistema de arquivos, atualização de dados centralizada
		3.0.0	13min31seg	–	2,96	13min31seg	40 Requisições, 3 workers, 3 processos, 4vCPUs, serviço de mensageria, atualização de dados descentralizada
Experimento 3	vCloud	2.11.6	17min15seg	18min23seg	3,47	35min38seg	60 Requisições, 2 workers, 3 processos,4vCPUs, sistema de arquivos, atualização de dados centralizada
		3.0.0	18min35seg	–	3,23	18min35seg	60 Requisições, 3 workers, 3 processos, 4vCPUs, serviço de mensageria, atualização de dados descentralizada

de CPU, visto que a carga de gerenciamento e de transporte de requisições ficou sobre o serviço de mensageria. O uso de memória RAM permaneceu praticamente o mesmo entre os componentes de ambas as versões, mantendo uma média de 3,6GB de uso.

Ao observar os resultados dos testes percebe-se que a versão 2.11.6 do Cloud.Jus realiza o provisionamento de recursos com aproximadamente o mesmo desempenho da versão 3.0.0, como pode ser observado no tempo de execução dos *workers* e nos valores de *throughput*. No entanto, é possível verificar que a versão 2.11.6 perde uma quantidade significativa de desempenho durante a atualização de dados de ambiente. Isso ocorre pelo fato da versão 2.11.6 utilizar arquivos para armazenar os dados de ambiente, como listas de VMs e *templates*, sendo que certos arquivos devem ser atualizados por completo, causando retrabalho e baixo desempenho.

Apesar da atualização de dados de ambiente não impedir a execução de requisições, visto que esta tarefa pode ser paralelizada utilizando novos processos, o alto tempo necessário para realizar a atualização causa assincronia entre os dados mostrados na interface de usuário e os dados reais. Desta forma, a versão 3.0.0 da plataforma, por conter um banco de dados robusto e eficiente, permite a atualização de dados de VMs específicas,

permitindo que os próprios módulos *workers* busquem os dados individuais da instância provisionada e enviem os dados atualizados da nova instância para a base de dados.

Além disso, Pode-se observar que o trabalho extra que os módulos *workers* da versão 3.0.0 possuem ao buscar os dados da nova instância provisionada e enviá-los à base de dados causa um leve impacto no desempenho e no *throughput* em comparação com o desempenho dos módulos *workers* da versão 2.11.6. No entanto, tal impacto pode ser diminuído ao paralelizar a execução das requisições ao utilizar mais módulos *workers*, visto que a carga de atualização de dados na versão 3.0.0 está sobre os módulos *workers*, enquanto na versão 2.11.6 o orquestrador *middleware* realiza a atualização dos dados. Adicionalmente, é possível perceber também que o tempo de execução aumentou de acordo com o número de requisições, no entanto, na versão 2.11.6 o tempo de atualização dos dados de ambiente permaneceu virtualmente o mesmo, mostrando que foi possível retirar uma carga fixa de processamento na versão 3.0.0, removendo o retrabalho e paralelizando a carga de atualização de dados, resultando em uma redução de aproximadamente 18 minutos e 13 segundos do tempo total por conjunto de requisições.

Diante do exposto, conclui-se que a versão 3.0.0 da plataforma Cloud.Jus introduziu melhorias significativas à solução, tornando seus componentes ainda mais modulares e desacoplados. Além disso, introduziu ferramentas e componentes que facilitaram o transporte, a atualização e o armazenamento de dados e requisições. Por fim, observou-se que as melhorias introduzidas também causaram um impacto positivo na eficiência e no desempenho da plataforma, diminuindo o retrabalho e facilitando a comunicação entre os componentes.

Capítulo 6

Conclusão

Neste trabalho foram apresentados os conceitos de computação em nuvem, além dos principais desafios ao realizar a implantação do modelo de nuvem em ambientes de infraestrutura legada. Para a resolução deste problema foi apresentada a plataforma de gerenciamento de nuvem Cloud.Jus [27]. O Cloud.Jus é uma solução desenvolvida e implantada no ambiente de infraestrutura legada do Supremo Tribunal Federal, entre 2017 e 2019, a fim de auxiliar a migração para um modelo de nuvem privada.

A plataforma Cloud.Jus, teve sucesso ao implantar um ambiente de computação em nuvem, integrando os hipervisores Hyper-V [98] e VMware [97] já e em funcionamento na infraestrutura legada do STF, com uma interface de usuário que permite o provisionamento de recursos sob demanda. No entanto, com o passar do tempo notou-se que os métodos de armazenamento e comunicação entre componentes escolhidos para a plataforma a tornavam pouco modular e consideravelmente acoplada. Diante disso, este trabalho propôs a implementação de melhorias no Cloud.Jus, como a adição de uma API e de um banco de dados para auxiliar no transporte de requisições e organização de dados. Também foi proposto a implantação do serviço de mensageria NATS [128] para realizar o gerenciamento de filas de requisições, a fim de facilitar o balanceamento de carga de execução das requisições realizadas pelos usuários da plataforma. Após a condução de experimentos e testes de estresse do sistema, foi possível concluir que houve ganho de eficiência após a adição de melhorias à plataforma, além de torna-lá mais modular e desacoplada.

Em relação à trabalhos futuros, percebe-se o potencial do Cloud.Jus em ser uma ferramenta capaz de ser exportada e implantada em outros órgãos do PJU. No entanto, antes que isso ocorra, é fundamental implementar uma camada de segurança mais robusta, adicionando criptografia SSL [159] no componente de API e implementando o uso de tokens para controlar o acesso às rotas de API. Adicionalmente, há planos de realizar a conteinerização dos componentes da plataforma e uma reformulação do sistema de

controle de acesso, a fim de desacopla-lo do modelo de controle de acesso específico do Supremo Tribunal Federal e torna-lo mais adaptável à outros ambientes. Dessa forma, a plataforma como um todo se tornaria desacoplada do ambiente do STF e estaria pronta para ser implantada em outros ambientes de nuvem dos órgãos do PJU, integrando suas infraestruturas em uma federação de nuvens comunitárias.

Referências

[1] *Magic quadrant for cloud infrastructure and platform services.* https://www.gartner.com/en/documents/4020235, acesso em 2023-05-01. ix, 9, 10, 13, 14

[2] *AWS global infrastructure - amazon web services.* https://aws.amazon.com/pt/about-aws/global-infrastructure/, acesso em 2023-04-30. ix, 10, 11

[3] *Explore | azure global infrastructure experience.* https://datacenters.microsoft.com/globe/explore, acesso em 2023-05-01. ix, 11, 12

[4] *Locais globais: regiões e zonas.* https://cloud.google.com/about/locations?hl=pt-br, acesso em 2023-05-01. ix, 13

[5] Toosi, Adel Nadjaran, Rodrigo N. Calheiros e Rajkumar Buyya: *Interconnected cloud computing environments: Challenges, taxonomy, and survey.* 47(1):7:1–7:47, ISSN 0360-0300. https://dl.acm.org/doi/10.1145/2593512, acesso em 2023-04-23. ix, 14, 15, 16, 49

[6] *Open source cloud computing platform software.* https://www.openstack.org/software/, acesso em 2023-06-10. ix, 18, 19, 20, 24

[7] *Home.* https://xenproject.org/, acesso em 2023-06-10. ix, 19, 24

[8] *OpenNebula – open source cloud & edge computing platform.* https://opennebula.io/, acesso em 2023-06-12. ix, 24, 25, 26

[9] Kumar, Rakesh, Kanishk Jain, Hitesh Maharwal, Neha Jain e Anjali Dadhich: *Apache CloudStack: Open source infrastructure as a service cloud computing platform.* ix, 18, 26, 28

[10] *Eucalyptus architecture overview :: Eucalyptus documentation.* https://docs.eucalyptus.cloud/eucalyptus/5/install_guide/eucalyptus/planning/euca_architecture/, acesso em 2023-06-15. ix, 29, 30

[11] Buyya, Rajkumar, James Broberg e Andrzej M. Goscinski: *Cloud Computing: Principles and Paradigms.* John Wiley & Sons, ISBN 978-1-118-00220-9. Google-Books-ID: S1NvRRd77rQC. ix, 32, 37

[12] Bittman, Thomas J, Mark A Margevicius e Philip Dawson: *Magic quadrant for x86 server virtualization infrastructure.* ix, 38

[13] *What is a container? | docker.* https://www.docker.com/resources/what-container/, acesso em 2023-06-22. ix, 39

[14] *O que é kubernetes?* https://www.redhat.com/pt-br/topics/containers/what-is-kubernetes, acesso em 2023-06-22. ix, 40, 41

[15] Quevedo, Waldemar: *Introduction to NATS.* Em Quevedo, Waldemar (editor): *Practical NATS: From Beginner to Pro*, páginas 1–18. Apress, ISBN 978-1-4842-3570-6. https://doi.org/10.1007/978-1-4842-3570-6_1, acesso em 2023-06-25. ix, 41, 42, 45

[16] *Publish-subscribe.* https://docs.nats.io/nats-concepts/core-nats/pubsub, acesso em 2023-06-26. ix, 42

[17] *AMQP 0-9-1 model explained — RabbitMQ.* https://www.rabbitmq.com/tutorials/amqp-concepts.html, acesso em 2023-06-26. ix, 43

[18] Kreps, Jay, Neha Narkhede e Jun Rao: *Kafka: a distributed messaging system for log processing.* ix, 44, 45

[19] Gracioli, Giovani, Murray Dunne e Sebastian Fischmeister: *A comparison of data streaming frameworks for anomaly detection in embedded systems.* ix, 45

[20] Edwards, D: *Introducing devops to the traditional enterprise.* InfoQueue/eMag Issue, 14, 2014. 1, 17, 48

[21] Archiveddocs: *Process and criteria for evaluating services-based integration technologies.* https://learn.microsoft.com/en-us/previous-versions/aa480046(v=msdn.10), acesso em 2023-07-12. 1

[22] Aceto, Giuseppe, Alessio Botta, Walter De Donato e Antonio Pescapè: *Cloud monitoring: A survey.* Computer Networks, 57(9):2093–2115, 2013. 1

[23] Murugesan, San e Irena Bojanova: *Encyclopedia of cloud computing.* Wiley Online Library, 2016. 1

[24] Buyya, Rajkumar, Christian Vecchiola e S. Thamarai Selvi: *Mastering Cloud Computing: Foundations and Applications Programming.* Newnes, ISBN 978-0-12-409539-7. Google-Books-ID: wqKkqHJhPJQC. 1, 8, 34, 67

[25] Mell, Peter e Timothy Grance: *The NIST definition of cloud computing.* 1, 5, 6, 7

[26] Joukov, Nikolai e Vladislav Shorokhov: *Cloud interoperability via quick enterprise applications re-builds.* Volume 2, páginas 575–580. SCITEPRESS, ISBN 978-989-8565-52-5. https://www.scitepress.org/PublishedPapers/2013/45022, acesso em 2023-06-16. 1, 31, 32, 47

[27] Castro, Klayton, Gabriel Macedo, Aletéia Araújo e Leonardo Carvalho: *Cloud.Jus: Architecture for Provisioning Infrastructure as a Service in the Government Sector*. Pages: 421. 2, 32, 36, 37, 46, 67, 73

[28] Douglas, F: *Parkhill. the challenge to computer utility. addison-wesley.* 1966. 4

[29] *ACM: Digital library: Communications of the ACM.* https://dl.acm.org/doi/fullHtml/10.1145/1721654.1721672, acesso em 2023-03-01. 4

[30] Kushida, Kenji E., Jonathan Murray e John Zysman: *Cloud computing: From scarcity to abundance.* 15(1):5–19, ISSN 1573-7012. https://doi.org/10.1007/s10842-014-0188-y, acesso em 2023-03-01. 4

[31] Hofmann, Paul e Dan Woods: *Cloud computing: The limits of public clouds for business applications.* 14(6):90–93, ISSN 1941-0131. Conference Name: IEEE Internet Computing. 5

[32] Alashhab, Ziyad R., Mohammed Anbar, Manmeet Mahinderjit Singh, Yu Beng Leau, Zaher Ali Al-Sai e Sami Abu Alhayja'a: *Impact of coronavirus pandemic crisis on technologies and cloud computing applications.* 19(1):100059, ISSN 1674-862X. https://www.sciencedirect.com/science/article/pii/S1674862X20300665, acesso em 2023-03-01. 5

[33] Wang, Lizhe, Jie Tao, Marcel Kunze, Alvaro Canales Castellanos, David Kramer e Wolfgang Karl: *Scientific cloud computing: Early definition and experience.* Em *2008 10th IEEE International Conference on High Performance Computing and Communications*, páginas 825–830. 5

[34] Duan, Yucong, Guohua Fu, Nianjun Zhou, Xiaobing Sun, Nanjangud C. Narendra e Bo Hu: *Everything as a service (XaaS) on the cloud: Origins, current and future trends.* Em *2015 IEEE 8th International Conference on Cloud Computing*, páginas 621–628. ISSN: 2159-6190. 7

[35] Abe, John Olorunfemi e Burak Berk Ustundaug: *A data as a service (DaaS) model for GPU-based data analytics.* http://arxiv.org/abs/1802.01639, acesso em 2023-03-29. 7

[36] *Citrix DaaS delivers secure virtual apps and desktops to any device - citrix.* https://www.citrix.com/products/citrix-daas/, acesso em 2023-03-29. 7

[37] Passerini, Katia, Ayman El Tarabishy e Karen Patten: *Information Technology for Small Business: Managing the Digital Enterprise.* Springer Science & Business Media, ISBN 978-1-4614-3040-7. Google-Books-ID: NsnQ8JxCpNIC. 7

[38] Faria, Heitor Medrado de: *A backup-as-a-service (BaaS) software solution.* https://repositorio.unb.br/handle/10482/34076, acesso em 2023-03-29, Accepted: 2019-02-27T17:17:47Z. 7

[39] Lanteri, Alessandro, Mark Esposito e Terence Tse: *From fintechs to banking as a service: global trends banks cannot ignore.* https://blogs.lse.ac.uk/businessreview/, acesso em 2023-06-03, Num Pages: 3 Publisher: London School of Economics and Political Science. 7

[40] *A placement architecture for a container as a service (CaaS) in a cloud environment | SpringerLink*. https://link.springer.com/article/10.1186/s13677-019-0131-1, acesso em 2023-03-29. 7

[41] Lehner, Wolfgang e Kai Uwe Sattler: *Database as a service (DBaaS)*. Em *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, páginas 1216–1217. ISSN: 2375-026X. 7

[42] Lynn, Theo, Pierangelo Rosati, Arnaud Lejeune e Vincent Emeakaroha: *A preliminary review of enterprise serverless cloud computing (function-as-a-service) platforms*. Em *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, páginas 162–169. ISSN: 2330-2186. 7

[43] Ribeiro, Mauro, Katarina Grolinger e Miriam A.M. Capretz: *MLaaS: Machine learning as a service*. Em *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, páginas 896–902. 7

[44] *IBM watson studio - overview*. https://www.ibm.com/cloud/watson-studio, acesso em 2023-03-29. 7

[45] infrastructure, Storage as a Serviceis the practice of using public cloud storage resources to store your data Using STaaS is more cost efficient than building private storage e Especially When You Can Match Data Types to Cloud Storage Offerings: *Storage as a service (STaaS)*. https://www.intel.com/content/www/us/en/cloud-computing/storage-as-a-service.html, acesso em 2023-03-29. 7

[46] *Dell APEX data storage services - armazenamento como serviço*. https://www.dell.com/pt-br/dt/apex/cloud-services/data-storage-services.htm, acesso em 2023-03-29. 7

[47] Zawoad, Shams, Amit Kumar Dutta e Ragib Hasan: *SecLaaS: secure logging-as-a-service for cloud forensics*. Em *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, ASIA CCS '13, páginas 219–230. Association for Computing Machinery, ISBN 978-1-4503-1767-2. https://dl.acm.org/doi/10.1145/2484313.2484342, acesso em 2023-03-29. 7

[48] Han, Lei: *Market acceptance of cloud computing - an empirical analysis of market structure, price models and service requirements*. https://epub.uni-bayreuth.de/id/eprint/524/, acesso em 2023-06-29, Place: Bayreuth Volume: 42. 8

[49] Colman-Meixner, Carlos, Chris Develder, Massimo Tornatore e Biswanath Mukherjee: *A survey on resiliency techniques in cloud computing infrastructures and applications*. 18(3):2244–2281, ISSN 1553-877X. Conference Name: IEEE Communications Surveys & Tutorials. 8

[50] Bittman, Thomas J, George J Weiss, Mark A Margevicius e Philip Dawson: *Magic quadrant for x86 server virtualization infrastructure*. 8, 38

[51] Malathi, M.: *Cloud computing concepts*. Em *2011 3rd International Conference on Electronics Computer Technology*, volume 6, páginas 236–239. 8

[52] *Magic quadrant research methodology.* https://www.gartner.com/en/research/methodologies/magic-quadrants-research, acesso em 2023-04-30. 9, 38

[53] *Serviços de computação em nuvem - amazon web services (AWS).* https://aws.amazon.com/pt/, acesso em 2023-04-30. 10

[54] Gupta, Bulbul, Pooja Mittal e Tabish Mufti: *A review on amazon web service (AWS), microsoft azure & google cloud platform (GCP) services.* ISBN 978-1-63190-292-5. https://eudl.eu/doi/10.4108/eai.27-2-2020.2303255, acesso em 2023-04-30. 10

[55] Dutta, Pranay e Prashant Dutta: *Comparative study of cloud services offered by amazon, microsoft & google.* 10

[56] *Soluções em nuvem por setor – amazon web services (AWS).* https://aws.amazon.com/pt/industries/, acesso em 2023-04-30. 10

[57] *Cloud solutions - soluções em nuvem - amazon web services.* https://aws.amazon.com/pt/solutions/, acesso em 2023-04-30. 10

[58] *Servidor privado virtual e hospedagem na web –amazon lightsail –amazon web services.* https://aws.amazon.com/pt/lightsail/, acesso em 2023-04-30. 10

[59] *Elastic compute cloud - amazon EC2 - AWS.* https://aws.amazon.com/pt/ec2/, acesso em 2023-04-30. 10, 23

[60] *Armazenamento s3 - simple storage service - amazon web services.* https://aws.amazon.com/pt/s3/, acesso em 2023-04-30. 10

[61] *AWS | amazon aurora - base de dados relacionais (RDBMS).* https://aws.amazon.com/pt/rds/aurora/, acesso em 2023-04-30. 11

[62] *Banco de dados de chave-valor NoSQL rápido – amazon DynamoDB – amazon web services.* https://aws.amazon.com/pt/dynamodb/, acesso em 2023-04-30. 11

[63] *Cloud computing services | microsoft azure.* https://azure.microsoft.com/en-us, acesso em 2023-05-01. 11, 38

[64] *OpenAI.* https://openai.com/, acesso em 2023-06-29. 11

[65] *Global infrastructure | microsoft azure.* https://azure.microsoft.com/en-us/explore/global-infrastructure, acesso em 2023-05-01. 11

[66] *Azure OpenAI service – advanced language models | microsoft azure.* https://azure.microsoft.com/en-us/products/cognitive-services/openai-service, acesso em 2023-05-01. 12

[67] *Azure SQL - family of SQL cloud databases | microsoft azure.* https://azure.microsoft.com/en-us/products/azure-sql, acesso em 2023-05-01. 12

[68] *Azure cognitive service for vision with OCR and AI | microsoft azure.* https://azure.microsoft.com/en-us/products/cognitive-services/vision-services, acesso em 2023-05-01. 12

[69] *Azure machine learning - ML as a service | microsoft azure.* https://azure.microsoft.com/en-us/products/machine-learning, acesso em 2023-05-01. 12

[70] *Azure container instances | microsoft azure.* https://azure.microsoft.com/en-us/products/container-instances, acesso em 2023-05-01. 12

[71] *Microsoft defender for cloud - CSPM & CWPP | microsoft azure.* https://azure.microsoft.com/en-us/products/defender-for-cloud, acesso em 2023-05-01. 12

[72] *Serviços de computação em nuvem.* https://cloud.google.com/?hl=pt-br, acesso em 2023-05-01. 12

[73] Machado, Ana Claudia Teixeira: *A FERRAMENTA GOOGLE DOCS: CONSTRUÇÃO DO CONHECIMENTO ATRAVÉS DA INTERAÇÃO e COLABORAÇÃO*. 12

[74] *Alibaba cloud: Cloud computing services*. https://www.alibabacloud.com, acesso em 2023-05-01. 13, 14

[75] *Discover the next generation cloud platform.* https://www.oracle.com/cloud-rh08/, acesso em 2023-05-01. 14

[76] *Tencent cloud*. https://www.tencentcloud.com/, acesso em 2023-05-01. 14

[77] *IBM cloud*. https://www.ibm.com/cloud, acesso em 2023-05-01. 14

[78] *Huawei cloud - everything-as-a-service | huawei cloud*. https://www.huaweicloud.com/intl/pt-br/, acesso em 2023-05-01. 14

[79] Grozev, Nikolay e Rajkumar Buyya: *Inter-cloud architectures and application brokering: taxonomy and survey.* 44(3):369–390, ISSN 1097-024X. https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2168, acesso em 2023-04-23, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.2168. 14, 15

[80] Assis, M. R. M. e L. F. Bittencourt: *A survey on cloud federation architectures: Identifying functional and non-functional properties.* 72:51–71, ISSN 1084-8045. https://www.sciencedirect.com/science/article/pii/S1084804516301436, acesso em 2023-04-23. 15

[81] Celesti, Antonio, Francesco Tusa, Massimo Villari e Antonio Puliafito: *How to enhance cloud architectures to enable cross-federation.* Em *2010 IEEE 3rd International Conference on Cloud Computing*, páginas 337–345. ISSN: 2159-6190. 15, 32

[82] Goiri, Inigo, Jordi Guitart e Jordi Torres: *Characterizing cloud federation for enhancing providers' profit.* Em *2010 IEEE 3rd International Conference on Cloud Computing*, páginas 123–130. ISSN: 2159-6190. 16

[83] Foster, Ian, Yong Zhao, Ioan Raicu e Shiyong Lu: *Cloud computing and grid computing 360-degree compared*. Em *2008 Grid Computing Environments Workshop*, páginas 1–10. ISSN: 2152-1093. 17

[84] Sosinsky, Barrie: *Cloud Computing Bible*. John Wiley & Sons, ISBN 978-1-118-02399-0. Google-Books-ID: aY4Kil7kbIcC. 17, 19, 41

[85] Sefraoui, Omar, Mohammed Aissaoui e Mohsine Eleuldj: *OpenStack: Toward an open-source solution for cloud computing*. 55:38–42. 18

[86] Milojičić, Dejan, Ignacio M. Llorente e Ruben S. Montero: *OpenNebula: A cloud management tool*. 15(2):11–14, ISSN 1941-0131. Conference Name: IEEE Internet Computing. 18, 24, 26

[87] Sotomayor, Borja, Rubén S. Montero, Ignacio M. Llorente e Ian Foster: *Virtual infrastructure management in private and hybrid clouds*. 13(5):14–22, ISSN 1941-0131. Conference Name: IEEE Internet Computing. 18, 35

[88] Freet, David, Rajeev Agrawal, Jessie J Walker e Youakim Badr: *Open source cloud management platforms and hypervisor technologies: A review and comparison*. Em *SoutheastCon 2016*, páginas 1–8. ISSN: 1558-058X. 18, 24

[89] Nurmi, Daniel, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff e Dmitrii Zagorodnov: *The eucalyptus open-source cloud-computing system*. Em *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, páginas 124–131. 18

[90] Bhardwaj, Tushar, Mohit Kumar e S. C. Sharma: *Megh: A private cloud provisioning various IaaS and SaaS*. Em Pant, Millie, Kanad Ray, Tarun K. Sharma, Sanyog Rawat e Anirban Bandyopadhyay (editores): *Soft Computing: Theories and Applications*, Advances in Intelligent Systems and Computing, páginas 485–494. Springer, ISBN 978-981-10-5699-4. 18, 33

[91] Munoz, Vıctor Mendez, Adrian Casajus Ramo, Ricardo Graciani Diaz e Andrei Tsaregorodtsev: *Cloud governance by a credit model with DIRAC*. 18

[92] Lima, Stanley, Álvaro Rocha e Licinio Roque: *An overview of OpenStack architecture: a message queuing services node*. 22(3):7087–7098, ISSN 1573-7543. `https://doi.org/10.1007/s10586-017-1034-x`, acesso em 2023-06-07. 18

[93] Jackson, Kevin e Cody Bunch: *OpenStack cloud computing cookbook: over 100 recipes to successfully set up manage your OpenStack cloud environments with complete coverage of Nova, Swift, Keystone, Glance, Horizon, Neutron, and Cinder*. Quick answers to common problems. Packt Publ, 2. aufl edição, ISBN 978-1-78216-758-7. 18, 19

[94] Shrivastwa, Alok, Sunil Sarat, Kevin Jackson, Cody Bunch, Egle Sigler e Tony Campbell: *OpenStack: Building a Cloud Environment*. Packt Publishing Ltd, ISBN 978-1-78712-941-2. Google-Books-ID: 7YJcDgAAQBAJ. 18, 19

[95] *NASA*. http://www.nasa.gov/index.html, acesso em 2023-06-29. 18

[96] *Rackspace technology | multicloud solutions provider*. https://www.rackspace.com/node/22215, acesso em 2023-06-29. 18

[97] *What is ESXI | bare metal hypervisor | ESX*. https://www.vmware.com/products/esxi-and-esx.html, acesso em 2023-06-10. 19, 37, 46, 73

[98] BenjaminArmstrong: *Hyper-v technology overview*. https://learn.microsoft.com/en-us/windows-server/virtualization/hyper-v/hyper-v-technology-overview, acesso em 2023-06-10. 19, 37, 38, 46, 73

[99] *KVM*. https://www.linux-kvm.org/page/Main_Page, acesso em 2023-06-10. 19

[100] *Open source cloud computing platform software*. https://www.openstack.org/software/, acesso em 2023-06-10. 24

[101] Narayana, K. e Jayashree Kanniappan: *A overview on cloud computing platforms and issues*. 7:238–242. 24

[102] *Apache cloudstack*. https://cloudstack.apache.org/, acesso em 2023-06-13. 26, 27, 28

[103] *Messaging that just works — RabbitMQ*. https://www.rabbitmq.com/, acesso em 2023-06-14. 29, 42, 43

[104] *Eucalyptus*. https://www.eucalyptus.cloud/, acesso em 2023-06-15. 29

[105] *Context-aware cloud application management:*. Em *Proceedings of the 4th International Conference on Cloud Computing and Services Science*, páginas 499–509. SCITEPRESS - Science and and Technology Publications, ISBN 978-989-758-019-2. http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0004949204990509, acesso em 2023-06-16. 31, 32

[106] Saatkamp, Karoline, Uwe Breitenbücher, Oliver Kopp e Frank Leymann: *Topology splitting and matching for multi-cloud deployments:*. Em *Proceedings of the 7th International Conference on Cloud Computing and Services Science*, páginas 275–286. SCITEPRESS - Science and Technology Publications, ISBN 978-989-758-243-1. http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006371002750286, acesso em 2023-06-16. 31

[107] Saldanha, Hugo Vasconcelos: *BioNimbus : uma arquitetura de federação de nuvens computacionais híbrida para a execução de workflows de bioinformática*. https://repositorio.unb.br/handle/10482/12046, acesso em 2023-06-16, Accepted: 2013-02-07T11:31:54Z. 32

[108] Mohammed, Bashir e Mariam Kiran: *Analysis of cloud test beds using OpenSource solutions*. Em *2015 3rd International Conference on Future Internet of Things and Cloud*, páginas 195–203. 33

[109] Maron, Carlos A F, Dalvan Griebler, Claudio Schepke e Luiz Gustavo Fernandes: *Desempenho de OpenStack e OpenNebula em estações de trabalho: Uma avaliação com microbenchmarks e NPB*. 33

[110] Endo, Patrícia Takako, Glauco Estácio Gonçalves, Judith Kelner e Djamel Sadok: *A survey on open-source cloud computing solutions*. 33

[111] Couto, Rodrigo S., Hugo Sadok, Pedro Cruz, Felipe F. da Silva, Tatiana Sciammarella, Miguel Elias M. Campista, Luís Henrique M. K. Costa, Pedro B. Velloso e Marcelo G. Rubinstein: *Building an IaaS cloud with droplets: a collaborative experience with OpenStack*. 117:59–71, ISSN 1084-8045. `https://www.sciencedirect.com/science/article/pii/S1084804518301929`, acesso em 2023-06-07. 33

[112] Razavi, Kaveh, Stefania Costache, Andrea Gardiman, Kees Verstoep e Thilo Kielmann: *Scaling vm deployment in an open source cloud stack*. Em *Proceedings of the 6th Workshop on Scientific Cloud Computing*, páginas 3–10, 2015. 34

[113] Singh, Sukhpal e Inderveer Chana: *A survey on resource scheduling in cloud computing: Issues and challenges*. 14(2):217–264, ISSN 1572-9184. `https://doi.org/10.1007/s10723-015-9359-2`, acesso em 2023-06-07. 34

[114] Da Cunha Rodrigues, Guilherme, Rodrigo N. Calheiros, Vinicius Tavares Guimaraes, Glederson Lessa dos Santos, Márcio Barbosa de Carvalho, Lisandro Zambenedetti Granville, Liane Margarida Rockenbach Tarouco e Rajkumar Buyya: *Monitoring of cloud computing environments: concepts, solutions, trends, and future directions*. Em *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, páginas 378–383. Association for Computing Machinery, ISBN 978-1-4503-3739-7. `https://doi.org/10.1145/2851613.2851619`, acesso em 2023-06-07. 34

[115] Puthal, Deepak, B.P.S. Sahoo, Sambit Mishra e Satyabrata Swain: *Cloud computing features, issues, and challenges: A big picture*. Em *2015 International Conference on Computational Intelligence and Networks*, páginas 116–123. ISSN: 2375-5822. 36

[116] Opara-Martins, Justice, Reza Sahandi e Feng Tian: *Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective*. Journal of Cloud Computing, 5:1–18, 2016. 36

[117] Kris, J: *Cloud computing: Saas, paas, iaas, virtualization, business models, mobile, security and more*. 2012. 37, 49

[118] Gu, Zonghua e Qingling Zhao: *A state-of-the-art survey on real-time issues in embedded systems virtualization*. 2012. `http://www.scirp.org/journal/PaperInformation.aspx?PaperID=18574`, acesso em 2023-06-20, Publisher: Scientific Research Publishing. 37

[119] Sahoo, Jyotiprakash, Subasish Mohapatra e Radha Lath: *Virtualization: A survey on concepts, taxonomy and associated security issues*. Em *2010 Second International Conference on Computer and Network Technology*, páginas 222–226. 37, 39

[120] Savill, John: *Azure stack*. Em *Microsoft Azure Infrastructure Services for Architects: Designing Cloud Solutions*, páginas 281–296. Wiley, ISBN 978-1-119-59653-0. https://ieeexplore.ieee.org/document/9822758, acesso em 2023-06-21, Conference Name: Microsoft Azure Infrastructure Services for Architects: Designing Cloud Solutions. 38

[121] Tarasov, Vasily, Dean Hildebrand, Geoff Kuenning e Erez Zadok: *Virtual machine workloads: The case for new benchmarks for NAS*. 39

[122] Dua, Rajdeep, A Reddy Raja e Dharmesh Kakadia: *Virtualization vs containerization to support PaaS*. Em *2014 IEEE International Conference on Cloud Engineering*, páginas 610–614. 39, 40

[123] *Red hat OpenShift | plataforma empresarial de containers kubernetes*. https://www.redhat.com/pt-br/technologies/cloud-computing/openshift, acesso em 2023-06-22. 40

[124] Ebert, Christof, Gorka Gallardo, Josune Hernantes e Nicolas Serrano: *DevOps*. 33(3):94–100, ISSN 1937-4194. Conference Name: IEEE Software. 41

[125] Coulouris, George, Jean Dollimore, Tim Kindberg e Gordon Blair: *Sistemas Distribuídos - 5ed: Conceitos e Projeto*. Bookman Editora, ISBN 978-85-8260-054-2. Google-Books-ID: 6WU3AgAAQBAJ. 41

[126] *Kafka versus RabbitMQ | proceedings of the 11th ACM international conference on distributed and event-based systems*. https://dl.acm.org/doi/abs/10.1145/3093742.3093908, acesso em 2023-06-26. 41, 43, 44

[127] *Apache kafka*. https://kafka.apache.org/, acesso em 2023-06-26. 42, 43

[128] *NATS.io*. https://nats.io/, acesso em 2023-06-26. 42, 46, 63, 73

[129] *Overview*. https://docs.nats.io/nats-concepts/overview, acesso em 2023-06-26. 42

[130] Hiraman, Bhole Rahul, Chapte Viresh M. e Karve Abhijeet C.: *A study of apache kafka in big data stream processing*. Em *2018 International Conference on Information , Communication, Engineering and Technology (ICICET)*, páginas 1–3. 44

[131] *Apache ZooKeeper*. https://zookeeper.apache.org/, acesso em 2023-06-26. 44

[132] Breitenbücher, Uwe, Tobias Binz, Oliver Kopp, Frank Leymann e Johannes Wettinger: *A modelling concept to integrate declarative and imperative cloud application provisioning technologies*. Em *International Conference on Cloud Computing and Services Science*, volume 2, páginas 487–496. SCITEPRESS, 2015. 47

[133] Bosch, Jan, Morven Gentleman, Christine Hofmeister, Juha Kuusela *et al.*: *Software architecture [electronic resource]: System design, development and maintenance*. 48

[134] *Eucalyptus, CloudStack, OpenStack and OpenNebula: A tale of two cloud models.* https://opennebula.io/eucalyptus-cloudstack-openstack-and-opennebula-a-tale-of-two-cloud-models/, acesso em 2023-06-12, Section: Blog. 48

[135] iainfoulds: *Active directory domain services overview.* https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview, acesso em 2023-07-05. 49

[136] Saffer, Dan: *Designing for interaction: creating innovative applications and devices.* New Riders, 2010. 49

[137] Dukarić, Robert e Matjaž B Jurič: *A taxonomy and survey of infrastructure-as-a-service systems.* Lecture Notes on Information Theory Vol, 1(1), 2013. 50, 65, 66

[138] sdwheeler: *Documentação do PowerShell - PowerShell.* https://learn.microsoft.com/pt-br/powershell/, acesso em 2023-07-06. 51, 61

[139] *PHP: Hypertext preprocessor.* https://www.php.net/index.php, acesso em 2023-07-06. 51, 53

[140] Silva, Mauricio Samy: *HTML5: a linguagem de marcação que revolucionou a web.* Novatec Editora, 2019. 51, 53

[141] Flanagan, David: *JavaScript: o guia definitivo.* Bookman Editora, 2004. 52, 58

[142] Pereira, Caio Ribeiro: *Aplicações web real-time com Node. js.* Editora Casa do Código, 2014. 52, 58

[143] Mumbaikar, Snehal, Puja Padiya *et al.*: *Web services based on soap and rest principles.* International Journal of Scientific and Research Publications, 3(5):1–4, 2013. 52

[144] *ArangoDB - home.* https://www.arangodb.com/, acesso em 2023-07-06. 52, 58

[145] *phpIPAM IPAM IP address management software.* https://phpipam.net/, acesso em 2023-07-07. 52, 65

[146] *Zabbix :: The enterprise-class open source network monitoring solution.* https://www.zabbix.com/, acesso em 2023-07-07. 52, 65

[147] *Grafana: The open observability platform.* https://grafana.com/, acesso em 2023-07-07. 52, 65

[148] Howes, Tim, Mark Smith e Gordon S Good: *Understanding and deploying LDAP directory services.* Addison-Wesley Professional, 2003. 53

[149] Efron, Bradley e Robert J Tibshirani: *An introduction to the bootstrap.* CRC press, 1994. 53

[150] *Gentelella alela! /.* https://colorlib.com/polygon/gentelella/, acesso em 2023-07-07. 53

[151] *Java | oracle.* https://www.java.com/pt-BR/, acesso em 2023-07-08. 58

[152] Satheesh, Mithun, Bruno Joseph D'mello e Jason Krol: *Web development with MongoDB and NodeJs.* Packt Publishing Ltd, 2015. 59

[153] *About npm | npm docs.* https://docs.npmjs.com/about-npm, acesso em 2023-07-08. 59

[154] *Instalando o express.* https://expressjs.com/pt-br/starter/installing.html, acesso em 2023-07-08. 59

[155] *Introduction to ArangoDB's technical documentation and ecosystem | ArangoDB documentation.* https://www.arangodb.com/docs/stable/, acesso em 2023-07-08. 59

[156] *JSON.* https://www.json.org/json-pt.html, acesso em 2023-07-08. 59, 60, 64, 67

[157] Josuttis, Nicolai M: *The c++ standard library: a tutorial and reference.* 2012. 59

[158] *Download VMware vSphere hypervisor (ESXi) 6.5.* https://customerconnect.vmware.com/downloads/get-download?downloadGroup=ESXI650, acesso em 2023-07-10. 70

[159] Wagner, David, Bruce Schneier *et al.*: *Analysis of the ssl 3.0 protocol.* Em *The Second USENIX Workshop on Electronic Commerce Proceedings*, volume 1, páginas 29–40, 1996. 73