



**Universidade Estadual do Sudoeste da Bahia – UESB**  
**Departamento de Ciências Exatas e Tecnológicas – DCET**  
**Curso de Ciência da Computação**

**Dênys Nepomuceno Batista**

**Estudo Comparativo dos serviços em nuvem Openshift e AWS Elastic Beanstalk**

**Vitória da Conquista**  
**2015**

**Dênys Nepomuceno Batista**

**Estudo Comparativo dos serviços em nuvem Openshift e AWS Elastic Beanstalk**

**Projeto de pesquisa da monografia de  
Conclusão de Curso de Ciência da  
Computação, na Universidade Estadual do  
Sudoeste da Bahia.**

**Orientador (a): Stenio Longo Araújo**

**Vitória da Conquista  
2015**

## **RESUMO**

Este trabalho tem por objetivo apresentar um estudo referente à Computação em nuvens e suas principais plataformas e comparar os serviços da plataforma Openshift e do Elastic Beanstalk da AWS, evidenciando as principais características de cada um. Através do referencial teórico explicamos o conceito geral e a história da computação em nuvens, destacando sua arquitetura, seus modelos e seus modos de implantação. Em seguida, foi apresentado um estudo comparativo entre os serviços em nuvem citados anteriormente, acrescentando a esse estudo a implantação de uma aplicação web de controle financeiro nos dois serviços. Através dos testes realizados, foi percebido que o Openshift apresentou maior velocidade de acesso e melhor desempenho para várias conexões por segundo.

Palavras-chave: Computação em nuvens. Openshift. AWS Elastic Beanstalk.

## **ABSTRACT**

This paper aims to present a study related to cloud computing and its major platforms and compare the services of OpenShift platform and AWS Elastic Beanstalk, highlighting the main features of each. Through the theoretical framework we explained the overall concept and the history of cloud computing, highlighting its architecture, its models and their deployment modes. Then, a comparative study was made between cloud services mentioned above, adding to this study the implementation of a financial control web application in both services. Through the tests, it was realized that OpenShift had lower access time and better performance for multiple connections per second.

**Keywords:** Cloud computing. Openshift. AWS Elastic Beanstalk.

## LISTA DE ABREVIATURAS E SIGLAS

API - *Application Programming Interface*  
APP - *Aplication*  
AWS - *Amazon Web Service*  
BD – *Banco de Dados*  
CPU - *Central Processing Unit*  
DNS - *Domain Name System*  
EC2 - *Elastic Compute Cloud*  
GB - *Giga Bytes*  
HTTP - *Hypertext Transfer Protocol*  
HTTPS - *Hypertext Transfer Protocol Secure*  
IAM - *Identity and Access Managemant*  
IIS - *Internet Information Services*  
IP - *Internet Protocol*  
IPv4 - *Internet Protocol versão 4*  
IS - *Information System*  
IT - *Information Technology*  
JDBC - *Java Database Connectivity*  
JDO - *Java Data Objects*  
JPA - *Java Persistence Application Programming Interface*  
JRE - *Java Runtime Environment*  
JSP - *Java sServer Pages*  
JVM - *Java Virtual Machine*  
MB - *Megabytes*  
PAAS - *Platform as a Service*  
PDA - *Pessonal Digital Assistant*  
PDF - *Portable Document Format*  
QoS - *Qualidade de Serviço*  
RDS - *Relational Database Service*  
SAAS - *Software as a Service*  
SDK - *Software Development Kit*  
SDL - *Security Development Lifecycle*  
SGBDR - *Sistema Gerenciador de Banco de Dados Relacional*  
SI - *Sistemas de Informação*  
SLA - *Service Level Agreement*  
SQL -*Structured Query Language*  
TI - *Tecnologia da Informação*  
TCO - *Total cost of ownership*  
URL - *Uniform Resource Locator*  
VHD - *Vitual Hard Disk*  
VM - *Virtual Machine*  
VMM - *Virtual Monitor Machine*  
VPN - *Virtual Private Network*  
WIF - *Windows Identity Foundation*

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO.....</b>	<b>7</b>
1.1	OBJETIVO GERAL.....	8
1.2	OBJETIVOS ESPECÍFICOS.....	8
1.3	METODOLOGIA.....	8
1.4	ESTRUTURA DO TRABALHO.....	8
<b>2</b>	<b>REFERENCIAL TEÓRICO.....</b>	<b>9</b>
2.1	ARQUITETURA DE COMPUTAÇÃO EM NUVEM.....	13
2.2	MODELO DE SERVIÇOS EM NUVEM.....	15
<b>2.2.1</b>	<b>Software como um Serviço (SAAS).....</b>	<b>15</b>
<b>2.2.2</b>	<b>Plataforma como um Serviço (PAAS).....</b>	<b>17</b>
<b>2.2.3</b>	<b>Infraestrutura como um Serviço (IAAS).....</b>	<b>18</b>
2.3	MODELOS DE IMPLANTAÇÃO EM NUVEM.....	19
2.4	VIRTUALIZAÇÃO.....	21
2.5	CATÁLOGO DE PLATAFORMAS EM NUVEM.....	22
<b>2.5.1</b>	<b>Openshift.....</b>	<b>22</b>
<b>2.5.2</b>	<b>Google AppEngine.....</b>	<b>24</b>
<b>2.5.3</b>	<b>Microsoft Azure.....</b>	<b>28</b>
<b>2.5.4</b>	<b>Amazon Web Services (AWS).....</b>	<b>34</b>
2.5.4.1	Amazon Elastic Compute Cloud (EC2).....	35
2.5.4.2	AWS Elastic Beanstalk.....	38
2.5.4.3	Amazon Simple Storage Service (Amazon S3).....	39
2.5.4.4	Amazon Relational Database Service (Amazon RDS).....	39
2.5.4.5	Amazon DynamoDB.....	40
2.6	TRABALHOS RELACIONADOS.....	41
<b>3</b>	<b>ESTUDO COMPARATIVO ENTRE OPENSIFT E ELASTIC BEANSTALK AWS.....</b>	<b>42</b>
3.1	CARACTERÍSTICAS DO OPENSIFT.....	43
<b>3.1.1</b>	<b>Linguagens e Servidores de Aplicação Compatíveis.....</b>	<b>44</b>
<b>3.1.2</b>	<b>Suporte a Banco de Dados.....</b>	<b>44</b>
<b>3.1.3</b>	<b>Segurança.....</b>	<b>44</b>
<b>3.1.4</b>	<b>Controle de Versão.....</b>	<b>45</b>

<b>3.1.5</b>	<b>Regiões Disponíveis Gratuitamente.....</b>	<b>45</b>
<b>3.2</b>	<b>CARACTERÍSTICAS DO AWS ELASTIC BEANSTALK.....</b>	<b>46</b>
<b>3.2.1</b>	<b>Linguagens e Servidores de Aplicação Compatíveis.....</b>	<b>47</b>
<b>3.2.2</b>	<b>Suporte a Banco de Dados.....</b>	<b>47</b>
<b>3.2.3</b>	<b>Segurança.....</b>	<b>48</b>
<b>3.2.4</b>	<b>Controle de Versão.....</b>	<b>48</b>
<b>3.2.5</b>	<b>Regiões Disponíveis Gratuitamente.....</b>	<b>49</b>
<b>3.3</b>	<b>CENÁRIOS E TESTES DE DESEMPENHO.....</b>	<b>49</b>
<b>3.3.1</b>	<b>Implantação do Projeto no OpenShift.....</b>	<b>49</b>
<b>3.3.2</b>	<b>Implantação do projeto no Elastic Beanstalk da AWS Amazon.....</b>	<b>55</b>
<b>3.3.3</b>	<b>Velocidade de acesso.....</b>	<b>60</b>
<b>3.3.4</b>	<b>Desempenho para Várias Requisições por Segundo.....</b>	<b>62</b>
<b>3.4</b>	<b>CONCLUSÃO DO ESTUDO COMPARATIVO.....</b>	<b>63</b>
<b>4</b>	<b>CONCLUSÃO.....</b>	<b>65</b>
<b>5</b>	<b>REFERÊNCIAS.....</b>	<b>66</b>

# 1 INTRODUÇÃO

A tecnologia é um dos principais instrumentos de que dispõem as empresas para alcançar competitividade. Consequentemente, acadêmicos, fabricantes de equipamentos e desenvolvedores de sistemas têm investido em diversas soluções para simplificar o acesso à informação pela Internet e colocaram seu foco para suprir esta demanda.

A fim de minimizar o desperdício de recursos computacionais e também de capital com atividades mantenedoras e renovação de licenças de *software*, propõe-se a utilização de um ambiente centralizado em “nuvem”, não suscetível à perda de informações e acessível de qualquer lugar do mundo, através de dispositivos conectados à internet. Esse conceito é denominado Computação em Nuvem ou *Cloud Computing*.

Na computação em nuvem, os recursos de TI são fornecidos como um serviço, permitindo aos usuários acessarem os serviços sem a necessidade de conhecimento sobre a tecnologia utilizada. Assim, os usuários e empresas passaram a acessar os serviços sob demanda e independente de localização, o que aumentou a quantidade de serviços disponíveis. O uso destes é cobrado de acordo com as diferentes políticas de tarifação para o usuário final.

Segundo Nubling (2011), a principal vantagem da computação em nuvem para as organizações é a economia em equipamentos, licenças de software, e suporte das infraestruturas informáticas, custos que são terceirizados a valores bastante atrativos, derivados da economia de escala que os fornecedores de serviços *Cloud* conseguem obter. Entretanto, ela tende a ir bem além e se tornar responsável por suportar o crescimento e garantir vantagens competitivas para os negócios.

Há a necessidade então, de exibir os benefícios proporcionados pela utilização de todas as facilidades que a tecnologia sugere, não só o armazenamento, o processamento ou o acesso às informações de forma não local e dinâmica, mas todas essas funcionalidades aplicadas ao mesmo tempo, com objetivo de gerar o crescimento e apoiar o controle das grandes corporações.

## 1.1 OBJETIVO GERAL



O objetivo geral desta pesquisa acadêmica é apresentar um estudo referente à Computação em nuvens e principais plataformas de computação em nuvens e comparar os serviços da plataforma Openshift e do AWS Elastic Beanstalk, evidenciando as suas principais características.

## 1.2 OBJETIVOS ESPECÍFICOS

- a) Identificar as principais características da computação em nuvem;
- b) Expor um estudo sobre as principais plataformas em nuvem;
- c) Apresentar a implantação de um projeto web nos serviços Openshift e Elastic Beanstalk AWS;
- d) Comparar a plataforma Openshift e o serviço AWS Elastic Beanstalk.

## 1.3 METODOLOGIA

O trabalho implica numa metodologia qualitativa de caráter bibliográfico, que consiste na utilização de referências teóricas já publicadas para análise e discussão da tecnologia, mas também de caráter comparativo. Para esse estudo foram usados como base monografias, livros, artigos e sites que possuem como objetivos investigar a computação nas nuvens e suas principais plataformas. Para o estudo comparativo, utilizou-se também da implantação de um aplicativo na plataforma Openshift e no serviço Elastic Beanstalk da plataforma AWS para exemplificar o uso desses serviços em nuvem e para realização de testes de desempenho.

## 1.4 ESTRUTURA DO TRABALHO

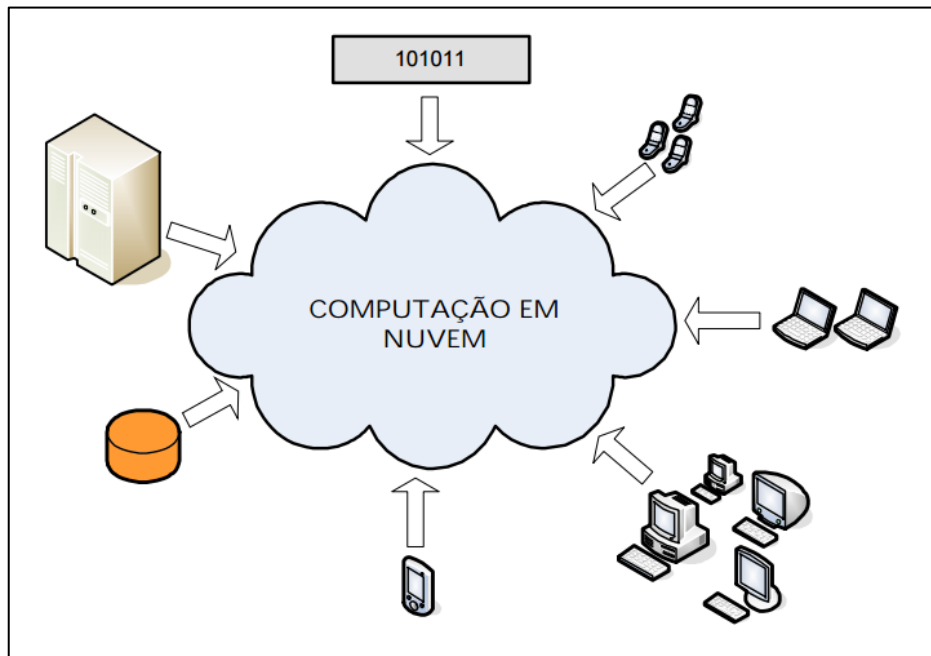
Para melhor visualização, o trabalho está estruturado em quatro capítulos, sendo estes: Capítulo I - introdução e apresentação geral do tema; Capítulo II - referencial teórico abordando os principais assuntos relativos ao tema, catálogo de plataformas e os trabalhos relacionados; Capítulo III - estudo comparativo e conclusões; Capítulo IV – Conclusão do trabalho.

## 2 REFERENCIAL TEÓRICO

Computação em Nuvem ou *Cloud Computing* é um tipo de sistema distribuído que consiste de um conjunto de computadores interconectados e usando a virtualização que é a subdivisão de um servidor físico em vários servidores lógicos. Esses computadores são provisionados dinamicamente e apresentados como um ou mais recursos computacionais em níveis de serviço estabelecidos através da negociação entre o provedor e seus consumidores.

Oliveira, Valentim e Santos (2011) sugerem que a ideia de computação em nuvem refere-se à utilização da memória e das capacidades de armazenamento e processamento de computadores compartilhados e interligados, seguindo o princípio da computação em grade. Esse conceito é realizado em serviços, os quais poderão ser acessados de qualquer lugar do mundo, a qualquer hora, sem a necessidade de armazenamento local de dados ou instalação de programas, acessando remotamente programas, arquivos e serviços através da Internet (ver Figura 1).

Figura 1 - Visão geral de uma nuvem computacional



Fonte: Ruschel, Zanotto e Mota, 2011

O termo Nuvem originou de uma metáfora sobre a Internet e sua combinação com computação – acesso a redes, elementos de armazenamentos, serviços de softwares, etc. Contudo, Computação em Nuvem não é um conceito inteiramente novo, sendo um conceito

que envolve conhecidas e maduras tecnologias como Computação em grade e Computação de alto desempenho.

Na década de 1960, John McCarthy, um importante pesquisador americano da área da informática e também um dos pioneiros da inteligência artificial, tratou de uma ideia bastante importante: computação por tempo compartilhado (*time sharing*), segundo a qual um computador pode ser utilizado simultaneamente por dois ou mais usuários para a realização de determinadas tarefas, aproveitando especialmente o intervalo de tempo ocioso entre cada processo (ALECRIM, 2008).

Segundo Nunes (2012), nessa mesma época, McCarthy propôs que a computação deveria ser organizada na forma de um serviço de utilidade pública, assim como os serviços de água e energia, em que os usuários só pagam pelo que usam, como acontece hoje nesse novo modelo de computação. Martins (2010) escreve que essa ideia era muito popular no final de década de 60. Entretanto, no meio da década de 70 ela foi abandonada quando se tornou claro que as tecnologias da informação da época não estavam aptas a sustentar um modelo desses de computação futurística.

As décadas de 1970 e 1980 foram marcadas pela centralização, segundo Anderson (2006). Quem quisesse ouvir uma música específica teria que escutar as rádios ou comprar um álbum inteiro do cantor. Existia aproximadamente meia dúzia de canais de televisão que centralizavam toda a programação e determinavam o que o público iria assistir ou saber. Nesta época, o modelo computacional também se baseava na centralização de informações através dos *mainframes*.

Ainda conforme Anderson (2006), a partir da década de 1990, com o surgimento da *Internet* como rede pública e outras tecnologias avançadas como a popularização da TV a cabo, o mundo migrou do mercado de massa para o mercado de nicho, aonde cada pessoa teria a seu dispor o conteúdo que quisesse. Agora, quem quisesse assistir tudo sobre esportes ou notícias teria acesso a canais exclusivos destes conteúdos. Quem quisesse ouvir uma música específica poderia comprá-la pela Internet ao invés de comprar o álbum inteiro.

Mesmo com a existência dessas ideias há tanto tempo, o termo computação em nuvem só veio a ser mencionado em 1997, numa palestra acadêmica do professor de sistemas da informação Ramnath Chellappa, e só foi desenvolvida no ano de 1999 com o surgimento da Salesforce.com, primeira empresa a disponibilizar aplicações na internet. A partir do sucesso dessa empresa, outras grandes começaram a investir na área, como a Amazon, a Google, a IBM e a Microsoft.

Conforme Lowe (2009), os criadores da Google, Sergey Brin e Larry Page, já haviam alcançado a capacidade da computação em nuvem durante a criação da infraestrutura da Google em 1997. Brin e Page utilizaram de diversos computadores convencionais que estavam sendo dispensados por sua universidade interligando-os uns aos outros para obter maior capacidade de armazenamento de dados (memória) junto com velocidade no processamento que resultaria em melhor tempo de resposta para qualquer pesquisa realizada na página da Google.

A ideia da computação em nuvem é exatamente igual. Se diversas máquinas forem adicionadas trabalhando simultaneamente é possível conseguir uma capacidade de processamento superior. Esta capacidade de processamento estando em um lugar público como a *Internet* (nuvem) e sendo vendido para organizações (mini nuvens) seria suficiente para que qualquer organização do século XXI fosse capaz de migrar todo ou quase todo o seu parque tecnológico para a mini nuvem.

Conforme Taurion (2009), a Computação em Nuvem é um fenômeno recente, mas que traz diversas vantagens competitivas para as organizações. Porém o *Cloud Computing* não está limitado a ser uma ferramenta somente para empresas, podendo ser usado para uso particular. Um exemplo desse uso seria o Google Docs, produto da Google que permite ao usuário particular salvar todos os seus dados como documentos, fotos, imagens, apresentações e planilhas em um servidor da Google na *Internet* que ficará disponível somente para o proprietário. Taurion (2009) informa que a computação em nuvem vai além de qualquer serviço de *hosting* (hospedagem de páginas na *Internet*).

Um usuário já está usufruindo da grande nuvem, se tão somente tenha uma conta em um servidor de *e-mail* ou um cadastro em alguma rede social, onde se armazena dados nesses servidores virtuais. Na realidade, uma simples troca está sendo feita, na qual se dispõem de serviços e tem-se espaço para guardar informações. Em contrapartida, são coletados os dados do usuário para um futuro uso comercial.

Através da publicação 800-145, o Instituto Nacional de Padrões e Tecnologias dos Estados Unidos (NIST) definiu que a computação em nuvem é um modelo de computação que oferece cinco essenciais características (Diogenes e Daniel, 2013):

- a) self-service sobre demanda: esta característica está ligada à capacidade de provisionamento de recursos de forma automatizada. Um exemplo disso seria a adição de mais servidores em um grupo existente de servidores para um determinado serviço.

- b) acesso amplo à rede: Esta característica está relacionada à capacidade de acesso, por parte de diversos recursos da rede através da manutenção da mesma experiência através de dispositivos, a recursos da rede computacional. Um exemplo disso seria o acesso a um recurso da rede através da manutenção da mesma experiência através de dispositivos distintos (um smartphone e o navegador de um computador pessoal);
- c) agrupamentos de recursos: Está relacionada à capacidade do provedor da nuvem de agrupar e mover recursos (físicos ou virtuais) para acomodar as necessidades de expansão e demanda do cliente. Podemos citar como exemplos de recursos os componentes básicos computacionais como memória, dispositivos de armazenamento, processador e rede;
- d) elasticidade rápida: refere-se à capacidade de rápido provisionamento de recursos de acordo com a demanda. Um exemplo disso seria um cliente que todo final de mês precisa de mais poder computacional que no restante do mês, e o provedor precisa dinamicamente alocar recursos para atender tal demanda do cliente;
- e) serviço mensurado: refere-se à capacidade de medir a utilização de recursos de acordo com o serviço oferecido. Esta é uma forma não só de monitorar, mas também de reportar os recursos em uso de uma forma transparente para o contratante do serviço.

Grandes empresas como Google, Amazon, Yahoo!, Facebook e Microsoft estão entre as companhias pioneiras, ofertando ampla gama de serviços e revolução do modelo em Nuvem, permitindo que suas aplicações sejam hospedadas e executadas remotamente em *datacenters*. Nesses cenários que surgem serviços populares com *e-mail* e redes sociais, os quais garantem acesso do usuário a alguns aplicativos – desde que tenha efetuado cadastro, dispensando armazenamento e processamento local nos computadores pessoais.

A computação em Nuvem tem sido usada como estratégia corporativa para otimizar o uso dos recursos computacionais da empresa. Esses recursos computacionais, que em épocas de pico encontram-se em funcionamento quase total, têm um custo justificado pelo processamento e hardware utilizado. Contudo, nos períodos de baixa demanda, os programas não ocupam todos os recursos a todo o momento. A computação nas nuvens traz grande vantagem para empresas nessas situações, com o propósito de buscar, mesmo na baixa demanda, uma melhor utilização dos equipamentos e recursos de tecnologia da informação.

A proposta da computação em nuvem é a virtualização de servidores ou parte fazendo com que a empresa somente utilize a capacidade que realmente está demandando, acessando os recursos de lugares diversos.

Há ainda algumas dificuldades a serem vencidas em sua totalidade na tecnologia de computação em nuvem. O provedor desta precisa modificar os mecanismos de provisionamento de custos, desempenho e segurança para os dados. Também deve haver constantes melhoras dos prestadores desse serviço para garantir integridade dos dados, confidencialidade e auditoria nos ambientes de máquinas virtuais, requisitos estes importantes para o controle e a garantia de execução de uma aplicação comercial. Além do mais, a falta de uma estrutura avançada de rede em muitos lugares se torna um grande obstáculo para uma completa disseminação deste tipo de serviço.

## 2.1 ARQUITETURA DE COMPUTAÇÃO EM NUVEM

A computação em nuvem possui uma arquitetura baseada em divisões lógicas de componentes de *hardware* e *software*, sendo que cada parte é uma camada que trata de uma particularidade na disponibilização de recursos para as aplicações (BUYYYA et al., 2009). Alguns destes recursos computacionais podem ser agrupados e organizados para realizar uma determinada tarefa do sistema como um todo. Cada camada pode ter seu gerenciamento ou monitoramento de forma independente das outras camadas, melhorando a flexibilidade, reusabilidade e escalabilidade no tocante a substituição ou adição de recursos computacionais sem afetar as outras camadas (SOUSA; MOREIRA; MACHADO, 2009).

A camada de mais baixo nível é a de infraestrutura física, representada por *datacenters*, *clusters*, *desktops* e outros recursos de *hardware*, podendo ter recursos heterogêneos. Com isso, fornece certa flexibilidade e facilidade de agregação de novos recursos à medida que se tornem necessários.

Acima da camada de de infraestrutura física há uma camada de *software* intermediária ou *middleware*, que gerencia a infraestrutura física e tem por objetivo fornecer um núcleo lógico de uma nuvem. Esta camada é representada por serviços de gerenciamento de virtualização, negociações de QoS (Qualidade de Serviço), serviços de cobrança, serviços para verificar aceitação de requisições baseado no QoS e preço, serviços para cálculo,

gerenciamento dos SLA (Acordo de Nível de Serviço), entre outros (SOUSA; MOREIRA; MACHADO, 2009).

No nível acima da camada de *middleware*, encontra-se a camada responsável por prover suporte para a construção de aplicações e que contem ferramentas ou ambientes de desenvolvimento. Estes ambientes possuem interfaces Web 2.0, *marshups*, componentes, recursos de programação concorrente e distribuída, suporte a *workflows*, bibliotecas de programação e linguagens de programação. Esta camada de desenvolvimento é utilizada pelos usuários mais experientes, aqueles que desenvolvem as soluções para computação em nuvem.

No nível mais alto encontra-se a camada das aplicações de computação em nuvem. Esta camada é de interesse do usuário, pois é por meio dela que ele utiliza os aplicativos. As camadas abaixo desta são responsáveis pelas características de escalabilidade, disponibilidade, ilusão de recursos infinitos e alto desempenho. Algumas soluções de arquitetura podem incluir uma camada de gerenciamento de adaptações sendo esta responsável por fornecer adaptação a estas soluções. Essas adaptações ocorrem de forma automática ou semi-automática e com isso, diminuem os esforços humanos para gerenciar arquiteturas de computação em nuvem. A Figura 2 ilustra as camadas citadas e suas respectivas associações.

**Figura 2: Arquitetura da Computação em Nuvem**

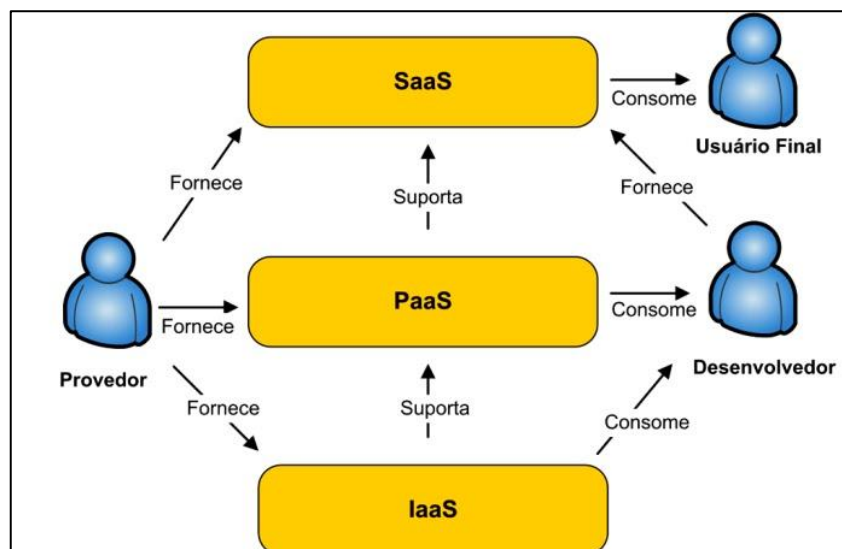


Fonte: adaptado de Vecchiola, Chu, e Buyya (2009).

## 2.2 MODELOS DE SERVIÇOS EM NUVEM

A computação em nuvem é descrita basicamente em três modelos: software como um serviço (SaaS), plataforma como um serviço (PaaS) e infraestrutura como um serviço (IaaS). O ideal é que estes serviços sejam disponibilizados para seus usuários desenvolvedores e/ou usuários finais de forma que os mesmos não tenham nenhuma responsabilidade quanto à infraestrutura física e monitoramento da nuvem. Os serviços podem interagir de forma que um fornece recursos para o outro ou podem ser consumidos individualmente.

**Figura 3: Modelos de serviço em computação em nuvem.**



Inter-relação entre fornecedores e consumidores.

Fonte: Vitti (2012).

Podemos ver alguns serviços de *Cloud Computing* já disponibilizados por algumas empresas dentro das três categorias no Quadro 1.

### 2.2.1 Software como um Serviço (SAAS)

Este talvez seja o modelo mais fácil de entender do ponto de vista de quem está adquirindo o serviço e provavelmente é o modelo que vai trazer um impacto maior na redução



**Quadro 1: Serviços já disponibilizados para Cloud Computing.**

<b>Infra-estrutura como um Serviço</b>	<b>Plataforma como um Serviço</b>	<b>Software como um Serviço</b>
<ul style="list-style-type: none"> <li>• Amazon EC2</li> <li>• Joyent</li> <li>• Sun            Microsoft's Networks.com</li> <li>• HP Flexible Computing Services</li> <li>• IBM Blue Cloud</li> <li>• 3tera</li> <li>• OpSource</li> </ul>	<ul style="list-style-type: none"> <li>• Bungee Lab's</li> <li>• Bungee</li> <li>• Connect</li> <li>• Etelos</li> <li>• Coghead</li> <li>• Google App Engine</li> <li>• HP Adaptive Infrastructure as a Service</li> <li>• Salesforce.com</li> </ul>	<ul style="list-style-type: none"> <li>• Oracle        SaaS platform</li> <li>• Salesforce    Sales Force</li> <li>• Automation NetSuite</li> <li>• Google Apps</li> <li>• Workday Human</li> <li>• Capital Management</li> </ul>

Fonte: Leavitt e Neal (2009)

de custos. O modelo de *Software* como um Serviço se baseia na ideia de fornecer ao consumidor um determinado serviço que está sendo operado e mantido na nuvem com execução no dispositivo do usuário. Dispositivo este que pode ser um computador pessoal, um telefone inteligente (*smartphone*), um *tablet*, entre outros (DIOGENES; MAUSER, 2013).

Na realidade o *software* como um serviço, na prática, já vem sendo usado há muitos anos com o advento do serviço de correio eletrônico por parte de provedores, como são o Hotmail, GMail e outros. Estes serviços estão na nuvem e o usuário final está consumindo o *software* fornecido por este serviço, que por sua vez é um serviço dedicado de correio eletrônico.

O uso de *software* como serviço para outras aplicações traz uma nova realidade para as empresas. Tradicionalmente as empresas adquirem uma licença de *software*, treinam seus usuários, planejar atualizações que porventura venham a existir (implantação de novas versões do aplicativo), realizam manutenção etc. O ciclo de vida do *software* licenciado, se comparado ao *software* como serviço, é bem distinto. Seguem abaixo algumas vantagens do uso do modelo SaaS (DIOGENES; MAUSER, 2013):

- a) redução de custo de licenciamento;
- b) abstração e redução do custo de manutenção do software;

- c) redução do custo de atualização do software;
- d) adoção mais acelerada de novas tecnologias.

Os SaaS são serviços de software disponíveis na internet para serem acessados por múltiplos usuários de qualquer lugar e a qualquer momento. Os computadores pessoais funcionam somente como meios de comunicação entre o sistema, que está na nuvem, e o usuário (SILVA, 2011). Os desenvolvedores se concentram em inovação e não na infraestrutura, levando ao desenvolvimento rápido de sistemas de software. O SaaS reduz custos na medida que se paga somente pelo que foi realmente usado (MARTINS, 2012).

### **2.2.2 Plataforma como um Serviço (PAAS)**

O PaaS é uma infraestrutura que serve para implementar e testar aplicações na nuvem. É um modelo de aplicação que fornece todos os recursos necessários para construir aplicativos e serviços diretamente da internet, sem precisar instalar ou baixar algum *software*.

Este modelo é mais flexível do ponto de vista de padronizações necessárias para a empresa. Se a empresa precisa, além do *software* como um serviço, de uma plataforma de desenvolvimento para as aplicações customizadas que ele pretender ter, este é o modelo ideal. Neste modelo, a empresa estará utilizando o conjunto de elementos oferecidos pelo provedor de soluções para desenvolvimento de *software* e padronizações dos serviços (DIOGENES; MAUSER, 2013).

Apesar de este modelo ser mais flexível, o nível de abstração da manutenção dos serviços também continua transparente. Com isso, o desenvolvedor não precisará se preocupar com a manutenção da plataforma operacional, ou seja, não é ele que vai se preocupar em atualizar a versão X para a versão Z. O contratante do serviço também não tem acesso aos componentes da infraestrutura de rede, ou seja, ele não terá controle sobre os switches, os dispositivos de armazenamento e sistemas operacionais em uso. O nível de abstração da plataforma da nuvem continua existindo (DIOGENES; MAUSER, 2013).

Entre as vantagens deste modelo podemos citar:

- a) o mesmo conjunto de vantagens do software como serviço;
- b) aumento no nível de customização da plataforma para adequar-se ao modelo do

negócio;

c) flexibilidade para desenvolvimento de *software* dentro de uma plataforma única, com linguagem de programação que tira proveito da infraestrutura de computação na nuvem.

Os desenvolvedores de *software* usam frequentemente a PaaS para desenvolver e testar novos programas. Os clientes pagam por um servidor virtual, mas na realidade compartilham a mesma máquina física com muitos outros clientes. Pode-se alugar espaço no servidor por tempo ou bloco de tempo.

### **2.2.3 Infraestrutura como um Serviço (IAAS)**

O IaaS é responsável por prover toda a infraestrutura necessária para PaaS e o SaaS. Torna mais fácil e acessível o fornecimento de recursos, tais como servidores, rede, aplicações de *software*, armazenamento e outros recursos de computação fundamentais para construir um ambiente sob demanda (sistemas operacionais e aplicativos). Em suma, IaaS permite que a organização terceirize toda a sua infraestrutura de TI para a Nuvem fornecedora de apoio de seu dia-a-dia. Para organizações menores esta é uma solução viável (RUSCHEL; ZANOTTO; MOTA, 2010).

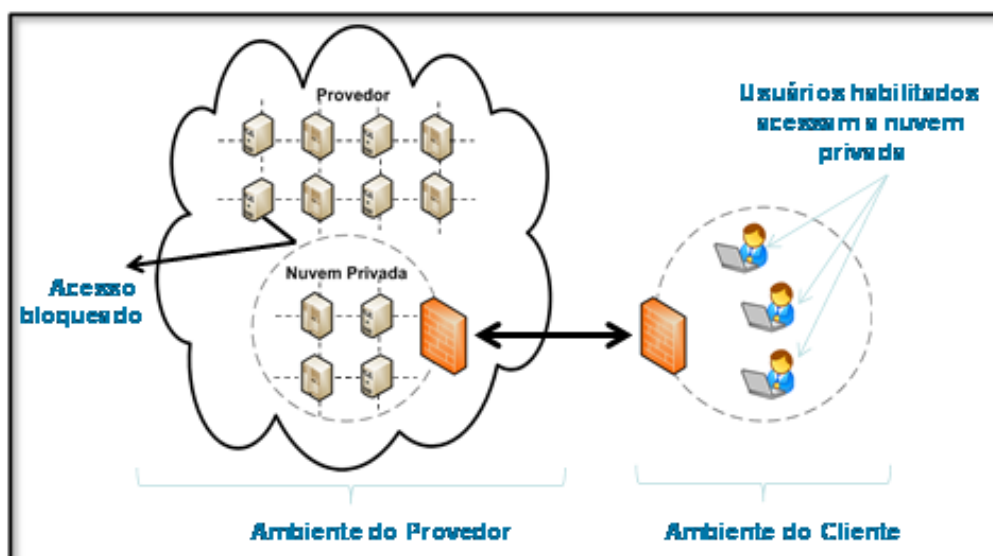
Nesse serviço, o consumidor não administra ou controla a infraestrutura de nuvem subjacente, mas tem controle sobre os sistemas operacionais, armazenamento, aplicativos implantados, e, eventualmente, o controle limitado de componentes de rede (por exemplo, *firewalls host*). O IaaS traz os serviços oferecidos na camada de infraestrutura, dentre os quais podemos incluir servidores, roteadores, sistemas de armazenamento e outros recursos de computação. O IaaS traz algumas características, como uma interface única para administração da infraestrutura, a aplicação API (*Application Programming Interface*) para interação com hosts, switches, roteadores e o suporte para a adicionar novos equipamentos de forma simples e transparente. O IaaS é baseado em técnicas de virtualização de recursos de computação. Observando do lado da economia, não será necessário a aquisição de novos servidores e equipamento de rede para a ampliação de serviços. (RUSCHEL; ZANOTTO; MOTA, 2010).

## 2.3 MODELOS DE IMPLANTAÇÃO EM NUVEM

É importante conceituar os modelos de implantação – compartilhados ou dedicados a hospedados interna ou externamente – sendo que seu formato é definido de acordo com a necessidade de aplicação. A restrição ou abertura de acesso depende do processo de negócios, do tipo de informação e do nível de visão desejado, se a organização deseja que todos os usuários possam ou não acessar e utilizar determinados recursos no seu ambiente de computação em nuvem. Os principais modelos de implantação são: nuvem privada, nuvem comunitária, nuvem pública e nuvem híbrida.

A nuvem privada é modelo utilizado unicamente por uma organização. Os serviços são oferecidos para serem utilizados internamente pela própria organização. Essa opção seria para empresas que desejam obter alta proteção dos dados e quando existe a necessidade de níveis mais rigorosos de privacidade ou de garantia de disponibilidade de aplicação. Pode ser gerenciado e operado pela equipe de TI interna ou através de um provedor externo que disponibiliza recursos exclusivos para a organização contratada com gerenciamento e operação (SÃO PAULO, 2015). Neste modelo de implantação são empregadas políticas de acesso aos serviços. As técnicas utilizadas para prover tais características podem ser em nível de gerenciamento de redes, configurações dos provedores de serviços e a utilização de tecnologias de autenticação e autorização (SOUSA; MOREIRA; MACHADO, 2009).

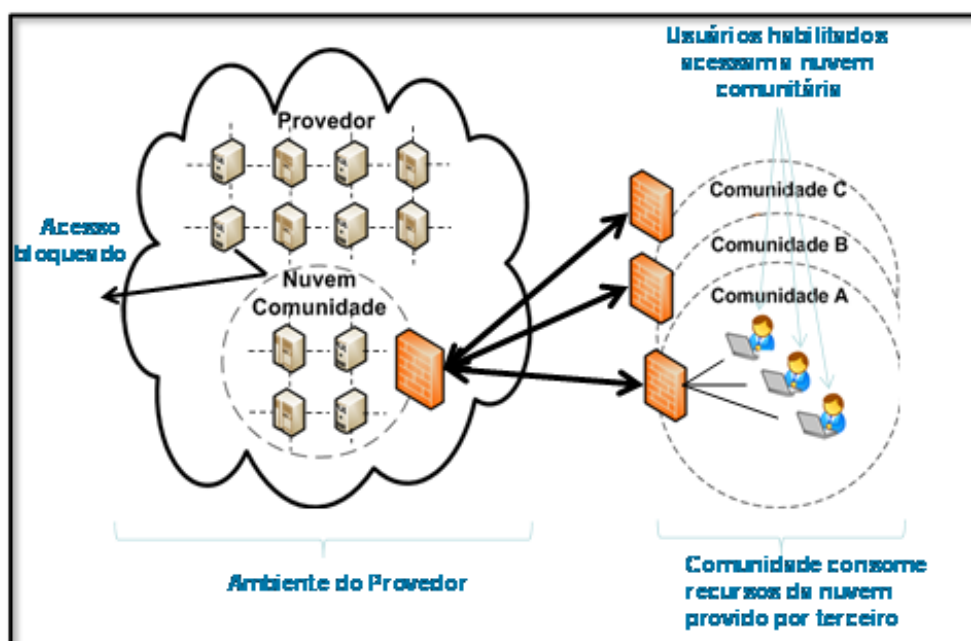
**Figura 4: Modelo de implantação nuvem privada provido por provedor terceiro.**



Fonte: São Paulo (2015).

A nuvem comunitária fornece uma infraestrutura compartilhada por uma comunidade de organizações com interesses em comum, que apresentam exigências semelhantes e decidem partilhar parte das suas infraestruturas. Essa opção oferece menor vantagem econômica do que a pública, porém pode oferecer um maior nível de segurança e privacidade. Pode ser administrada pelas próprias organizações ou por um terceiro e pode existir no ambiente da empresa ou fora dele.

**Figura 5: Modelo de implantação nuvem comunitária, provido por provedor terceiro.**

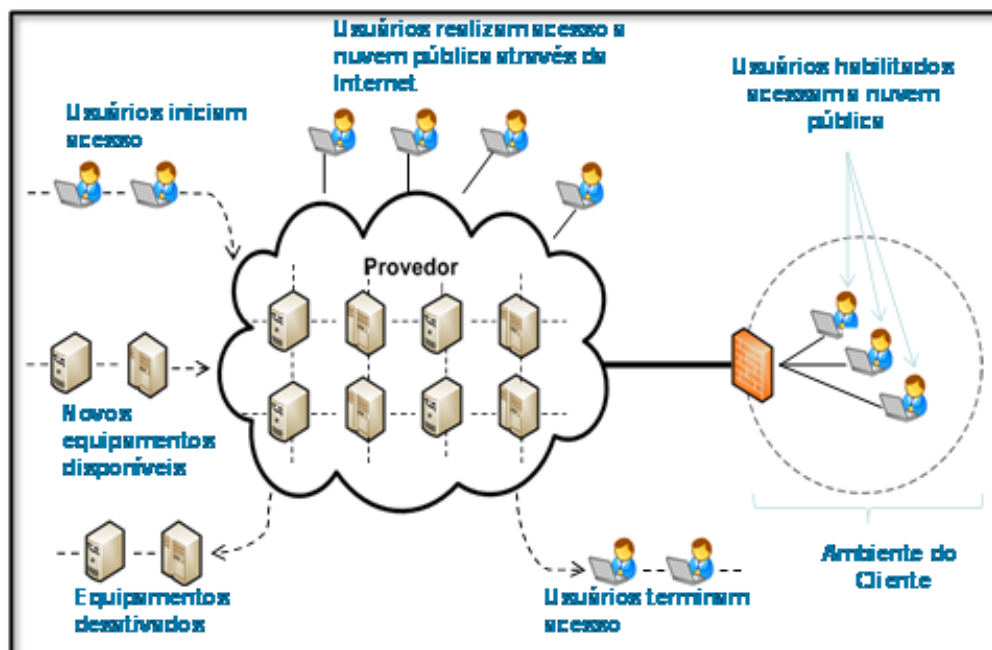


Fonte: São Paulo (2015).

No modelo de nuvem Pública a infraestrutura é aberta ao público em geral. Atualmente, na maioria dos casos, esse modelo funciona com a cobrança de serviço pelo seu uso real durante um período e são oferecidas por companhias que possuem grandes capacidades de armazenamento e processamento, onde os dados de diferentes contratantes podem ser salvos em diferentes servidores. É provido por um provedor terceiro que pode ser uma organização, comunidade acadêmica ou uma organização do governo, ou uma combinação deles (SÃO PAULO, 2015).

Na implantação Híbrida ocorre a composição de duas ou mais Nuvens (privada, comunitária ou pública) que continuam a ser entidades únicas, porém conectadas através de tecnologia proprietária ou padronizadas (SOUSA; MOREIRA; MACHADO, 2009).

Figura 6: Modelo de implantação nuvem pública.



Fonte: São Paulo (2015).

## 2.4 VIRTUALIZAÇÃO

A virtualização é uma tecnologia que foi criada na década de 1960, pela empresa IBM, com a intenção inicial de ser usada em *Mainframes*, na tentativa de compartilhar porções menores de grandes e caros equipamentos. Pesquisas realizadas naquela década proporcionaram tal tecnologia de forma que, atualmente, ela tem tomado maiores proporções, podendo ser usada em equipamentos de diferentes tamanhos e se tornando uma das singularidades mais importantes atualmente na tecnologia da informação (TOSADORE, 2012).

É a subdivisão de um servidor físico em vários servidores lógicos através de um software de virtualização que cria uma camada de abstração entre o *hardware* e os sistemas operacionais de cada servidor lógico criado. Esta camada de abstração oferece para os sistemas operacionais instalados instruções de máquinas equivalentes ao processador físico (PINELI;DUARTE, 2015).

Com a virtualização, em um único servidor físico é possível ter diversas máquinas virtuais com diferentes sistemas operacionais e aplicações. Permite que empresas utilizem seus servidores com maior eficiência. Com a camada de abstração da virtualização, uma

máquina virtual e seus sistemas instalados passaram a ser um único arquivo que pode ser movido para outros hardwares, duplicado de acordo com a necessidade de demanda. Isto tudo exige uma infraestrutura que pode ser desde a mais simples até a mais complexa exigida na Computação em Nuvem (VERAS, 2012).

A utilização de virtualização traz economia em longo prazo, visto que teremos menor consumo eminentes de energia elétrica e arrefecimento do ambiente, além da depreciação com manutenção e aquisição de *hardware*. De um modo geral, a virtualização é uma combinação de engenharia de *software* e *hardware* que é capaz de criar máquinas virtuais (VMs), uma abstração de hardware de computador que permite uma única máquina atuar com se fosse varias máquinas distintas.

*Virtual Monitor Machine* (VMM) - em português monitores de máquinas virtuais- ou *hypervisor* surgiram para solucionar as desvantagens das máquinas virtuais. Os monitores são colocados entre o *hardware* e o sistema operacional, disponibilizando uma máquina virtual para o mesmo. Sendo assim, eles reconhecem e descobrem com a máxima eficiência os dispositivos físicos de entrada e saída.

## 2.5 CATÁLOGO DE PLATAFORMAS EM NUVEM

Nessa seção apresentaremos algumas plataformas que oferecem serviços no modelo PaaS como OpenShift da Red Hat Cloud Computing, Windows Azure da Microsoft, App Engine da Google e AWS da Amazon.

### 2.5.1 Openshift

Para disponibilizar serviços no ambiente em nuvem, é necessário que haja uma infraestrutura de servidores robusta e boa ambientação. Aproveitando a experiência que já possui no ramo, a *Red Hat* lançou, em maio de 2011, durante o evento '*Red Hat Summit*', o OpenShift (ver Figura 7) , uma plataforma como um serviço em nuvem que automatiza a hospedagem, a configuração, a implantação e a administração de aplicativos em um ambiente

flexível em nuvem. Ela cuida de toda a infraestrutura, *middleware* e gerenciamento de rede, permitindo que os desenvolvedores se concentrem apenas na criação de seus aplicativos.

**Figura 7: Página inicial do site da Openshift**



Fonte: Print Screen do site no navegador Google Chrome.

O modelo de solução tecnológica criado pela *Red Hat* suporta a implantação de *softwares* escritos em diversas linguagens como Java, Ruby, PHP, Perl, Python e Node.js. É oferecida ainda a possibilidade de uso de uma vasta gama de servidores, bancos de dados e *frameworks*. (OPENSIFT, 2015).

O usuário pode criar aplicativos na plataforma OpenShift por web console, linha de comando ou através de IDE, como por exemplo o Eclipse. Entretanto, a empresa orienta que deve ser usada pelo menos as ferramentas do cliente OpenShift (RHC), que deverão ser instaladas na máquina local.

Uma das principais vantagens na utilização do modelo OpenShift é que este permite ao desenvolvedor manter o foco no desenvolvimento da aplicação, realizando a implementação das aplicações de forma ágil, por meio de ferramentas disponibilizadas e possibilitando testá-las em um ambiente real de produção. É possível, ainda, mostrar, distribuir e compartilhar a aplicação funcionando sem se preocupar com a infraestrutura necessária. Tais fatores ajudam com a redução dos custos no projeto e no impacto ambiental dos recursos computacionais usados com a infraestrutura local (DIAS; JUNIOR, 2012).

Outra grande característica da plataforma OpenShift é o gerenciamento do volume de conexões aos aplicativos implantados. Ela faz escalonamento automático ou manual dos



recursos que apoiam as aplicações de modo que o desempenho não seja prejudicado à medida que varia o uso.

O OpenShift está disponível em três versões: “OpenShift *Online*”, “OpenShift *Enterprise*” e “OpenShift *Origin*” (OpenShift, 2015):

- a) OpenShift *Online*: é a versão gratuita da plataforma, que permite aos usuários criarem até três aplicações sem custos, permitindo utilizar por aplicação até 1GB de espaço em disco e 512 MB de memória. Nesta versão o usuário pode optar por outros planos em que se paga para utilizar recursos extras. O suporte é oferecido pela comunidade OpenShift, com exceção do plano Silver, onde o usuário poderá contar com a assistência da empresa *Red Hat*;
- b) OpenShift *Enterprise*: é a versão empresarial, com assinatura de software anual, permite ser implementada em uma nuvem privada. Essa versão permite acelerar a entrega de serviços e agilizar o desenvolvimento de aplicações, oferecendo um maior grau de controle e escolha sobre os componentes permitindo alocar mais recursos de hardware. O suporte técnico oferecido é profissional da empresa RedHat;
- c) OpenShift *Origin*: é voltado para a comunidade “*Open Source*”, onde está disponível todo o seu código fonte para ser copiado, possibilitando ao usuário, criar a sua própria versão do OpenShift localmente.

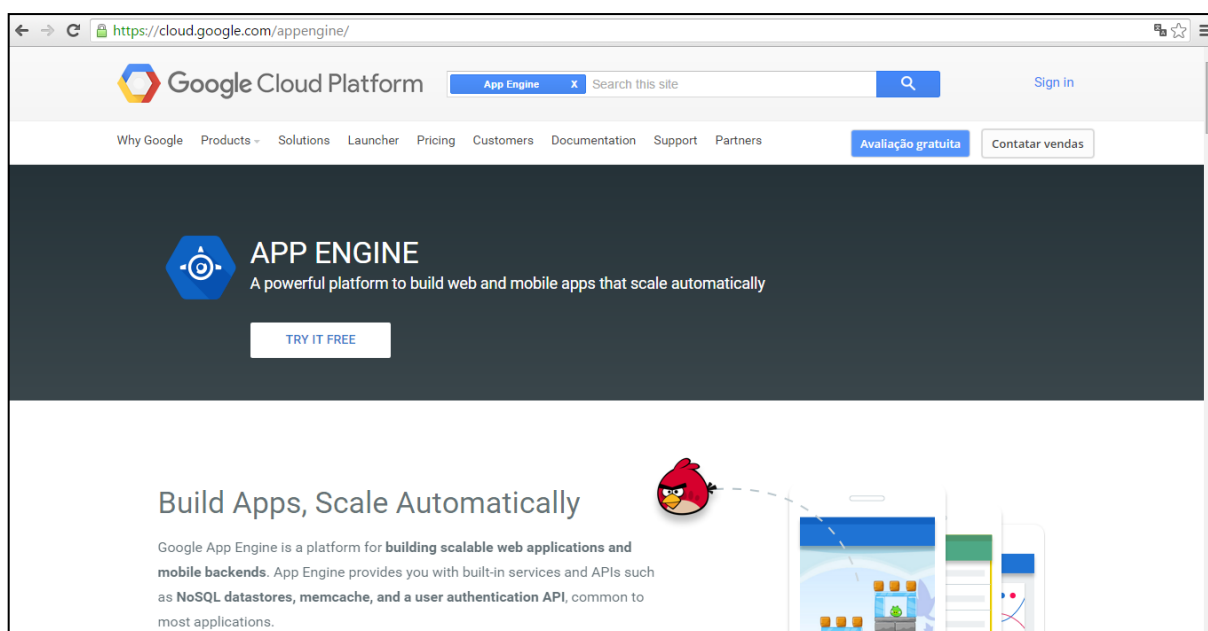
### **2.5.2 Google AppEngine**

Google *App Engine* (GAE) é uma plataforma como um Serviço (PaaS) disponibilizado pela empresa Google inc. (ver Figura 8), sobre a qual é possível desenvolver e hospedar aplicações públicas ou privadas nos servidores da mesma, tornando a criação facilitada e escalável conforme o armazenamento e tráfego aumentam ou diminuem. Ela fornece um conjunto de APIs e um modelo de aplicação que permite aos desenvolvedores utilizarem serviços adicionais fornecidos pelo Google, como o e-mail, armazenamento, entre outros. (SOUSA; MOREIRA, MACHADO, 2009).

A primeira versão da GAE foi lançada em Abril de 2008 e só permitia o desenvolvimento de *softwares* na linguagem de programação Python. Um ano depois, a

Google começava a disponibilizar sua plataforma agora com a JVM (Java *Virtual Machine*) (HOLLANDA, 2010). Desde então, a Google *App Engine* vem evoluindo e conquistando vários adeptos. Atualmente a GAE suporta aplicativos escritos nas linguagens Java, Python, PHP e Go, contudo os modelos de site pode incluir JavaScript , juntamente com o seu HTML que, entre outras coisas , permite escrever aplicações web AJAX. É possível também o uso de alguns frameworks como Flask, Django, and Bottle (SILVA, 2015).

**Figura 8: Página inicial da plataforma App Engine**



Fonte: Print Screen do site no navegador Google Chrome.

Para começar a utilização do Google *App Engine* é gratuito, podendo o usuário criar até 25 aplicativos gratuitos e utilizar até 1 GB de armazenamento de dados e CPU e largura de banda que sejam suficientes para suportar um aplicativo eficiente que ofereça cerca de cinco milhões de visualizações de página por mês. Como as demais PaaS, o GAE tem seu pagamento por utilização. Como se pode controlar a quantidade máxima de recursos disponibilizados pelos aplicativos, torna-se fácil que o aplicativo fique dentro das possibilidades de gasto. Após a ativação do faturamento do aplicativo, os limites gratuitos tornam-se maiores e desta forma paga-se apenas pelos recursos que ultrapassem a cota gratuita (GOOGLE, 2015).

O Google *App Engine* possui vários recursos como (PESSINI; MARTINHO; MAZZOLA, 2012):

- a) serviço da *web* dinâmico com suporte completo a tecnologias da *web* comuns;
- b) armazenamento persistente com consultas, classificação e transações;
- c) escalonamento automático e balanceamento de carga;
- d) um ambiente de desenvolvimento local com todos os recursos, simulando o Google App Engine em um computador local;
- e) filas de tarefas para realizar trabalho fora do escopo de uma solicitação da *web*;
- f) tarefas programadas para iniciar eventos em horários específicos e em intervalos regulares;
- g) integração com outros serviços em nuvem do Google e APIs.

Alguns recursos impõem limites para proteger a estabilidade do sistema que não estão relacionados a cotas, por exemplo quando um aplicativo é chamado para servir uma solicitação da *web*, ele deve emitir uma resposta em até 30 segundos, caso demore, o processo será encerrado e o servidor retornará um código de erro ao usuário, esse tempo de espera de solicitação é dinâmico e pode ser reduzido para poupar os recursos caso um manipulador de solicitação chegue ao tempo limite com muita frequência (GOOGLE, 2015).

Para gerenciar aplicativos executados no Google *App Engine*, deve ser utilizado o *Admin Console*. Este é uma interface baseada na *web* e é usada para criar novos aplicativos, configurar nomes de domínio, alterar a versão ativa de aplicativos, examinar registros de acessos e de erros e navegar pelo armazenamento de dados de um aplicativo (PESSINI; MARTINHO; MAZZOLA, 2012).

Para desenvolver e enviar aplicativos para a nuvem, o usuário da GAE deve usar os Kits de Desenvolvimento de *Software* (SDKs), que são disponibilizados para todas as linguagens suportadas pela plataforma. Cada SDK possui todas as APIs e bibliotecas disponíveis para *App Engine*, um simulado e seguro ambiente *sandbox* - que emula todos os serviços do Google *App Engine* no computador local - e ferramentas de implantação que permitem enviar o aplicativo para a nuvem e gerenciar diferentes versões deste (GOOGLE, 2015).

No *Sandbox* os aplicativos são executados em um ambiente seguro que fornece acesso limitado ao sistema operacional. Limitações como estas permitem que o Google *App Engine* distribua solicitações de *web* para o aplicativo entre diversos servidores, iniciando e interrompendo servidores para atender às demandas de tráfego.

O ambiente *App Engine* fornece várias opções para armazenar dados para a sua aplicação (GOOGLE, 2015):

- a) *App Engine Datastore*: Um armazenamento de dados objeto não esquematizado com cache automático, um mecanismo de consulta sofisticado, e transações atômicas;
- b) *Google Cloud SQL*: Um banco de dados relacional SQL para seu aplicativo *App Engine*, com base no familiar banco de dados MySQL;
- c) *Google Cloud Storage*: Um serviço de armazenamento de objetos e arquivos até terabytes de tamanho, e acessível à *App Engine* aplicativos através da biblioteca cliente do *Google Cloud Storage*.

O *Google App Engine* fornece um serviço de armazenamento de dados distribuído que contém um mecanismo de consultas e transações. Dessa forma, o servidor *web* distribuído cresce proporcionalmente ao tráfego (GOOGLE, 2015).

O armazenamento de dados é implementado nas transações por toda a sua rede distribuída usando "grupos de entidades". Dessa forma, as transações manipulam entidades dentro apenas de um grupo e essas entidades são armazenadas juntas para que a transação tenha total eficácia. Quando se cria um aplicativo, pode-se atribuir entidades para que a transação seja de total eficácia (GOOGLE, 2015).

O Google disponibiliza *plugin* para o IDE Eclipse que adiciona novos assistentes de projeto e configurações de depuração de projetos da *App Engine*. *App Engine* for Java torna especialmente fácil de desenvolver e implantar aplicativos web de classe mundial usando o *Google Web Toolkit* (GWT), que é um conjunto de ferramentas específico para desenvolvimento web, que facilita algumas tarefas usuais do desenvolvedor. O *plugin* Eclipse vem com os SDKs *App Engine* e GWT (SOARES, 2009).

Para o armazenamento de dados do *Google App Engine*, o SDK para Java inclui implementações das interfaces JDO e JPA. Os aplicativos podem utilizar a API JavaMail para enviar mensagens de e-mail com o serviço de e-mail do *Google App Engine*. As APIs HTTP *java.net* acessam o serviço de busca de URL do *Google App Engine*, que incluem APIs de nível inferior para implementar adaptadores adicionais ou para serem usadas diretamente do aplicativo. A utilização de compiladores ou interpretadores compatíveis com a JVM possibilita desenvolver aplicativos da web em outras linguagens, como JavaScript, Ruby ou Scala (PESSINI; MARTINHO; MAZZOLA, 2012).

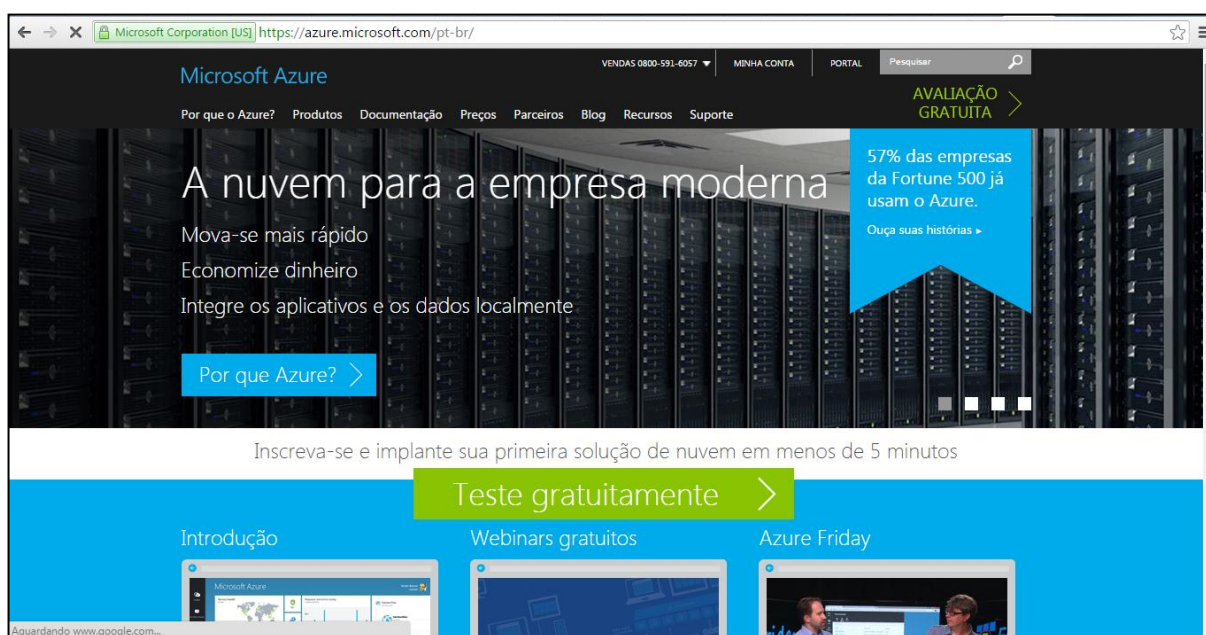
A GAE possibilita a integração de um aplicativo com as contas do Google para autenticar um usuário, que lhe permite que fazer login com uma conta do Google, possibilitando que o usuário comece a usar aplicativos com mais rapidez e sem a necessidade

de criação de uma nova conta. A API facilita também a implementação de áreas restritas a determinados tipos de usuários, já que é possível informar ao aplicativo se o usuário atual é um administrador registrado (GOOGLE, 2015).

### 2.5.3 Microsoft Azure

A Azure é uma plataforma de computação em nuvem da Microsoft (ver Figura 9), uma coleção crescente de serviços integrados - análise, computação, banco de dados, serviços móveis, redes, armazenamento e *Web*. A plataforma foi apresentada para os desenvolvedores e para o público no dia 27 de outubro de 2008 durante a Conferência de Desenvolvedores Profissionais da Microsoft, ocorrida em Los Angeles.

**Figura 9: Página inicial da plataforma Microsoft Azure**



Fonte: Print Screen do site no navegador Google Chrome.

A Microsoft Azure apresenta vários serviços dentro das categorias de serviços da computação em nuvem (IaaS, PaaS e SaaS). Esta plataforma pode ser utilizada de diversas maneiras, tais como construir uma aplicação *web* que executa e armazena seus dados em *datacenters* da Microsoft. Da mesma forma pode ser utilizada apenas para armazenamento de dados acessados por aplicações instaladas nos dispositivos computacionais do usuário, assim

como para criar máquinas virtuais ou ainda criar grandes aplicações altamente escaláveis com muitos usuários (MICROSOFT, 2015).

O sistema Azure é executado em uma rede mundial de data centers gerenciados pela Microsoft em 19 regiões – mais países e regiões do que os do Amazon *Web Services* e Google Cloud combinados. Azure é também o primeiro provedor de nuvem multinacional na China continental (MICROSOFT, 2015).

A plataforma da Microsoft dá suporte à mais ampla seleção de sistemas operacionais, linguagens de programação, frameworks, ferramentas, bancos de dados e dispositivos. É possível executar contêineres Linux com integração com Docker; compilar aplicativos com JavaScript, Python, .NET, PHP, Java, Node.js; compilar *back-ends* para dispositivos iOS, Android e Windows (MICROSOFT, 2015).

A Microsoft Azure oferece várias maneiras de hospedar sites da Web conforme ilustrado no quadro 2: Serviço de Aplicativo do Azure, Serviços de nuvem e Máquinas virtuais. Eles fornecem um conjunto de serviços diferente, portanto, qual deles deve ser escolhido depende exatamente do que o usuário está tentando desenvolver (DAVIES, 2015). A Figura 10 ilustra o grau relativo de controle em relação à facilidade de uso de cada uma dessas opções de hospedagem na *Web* no Azure.

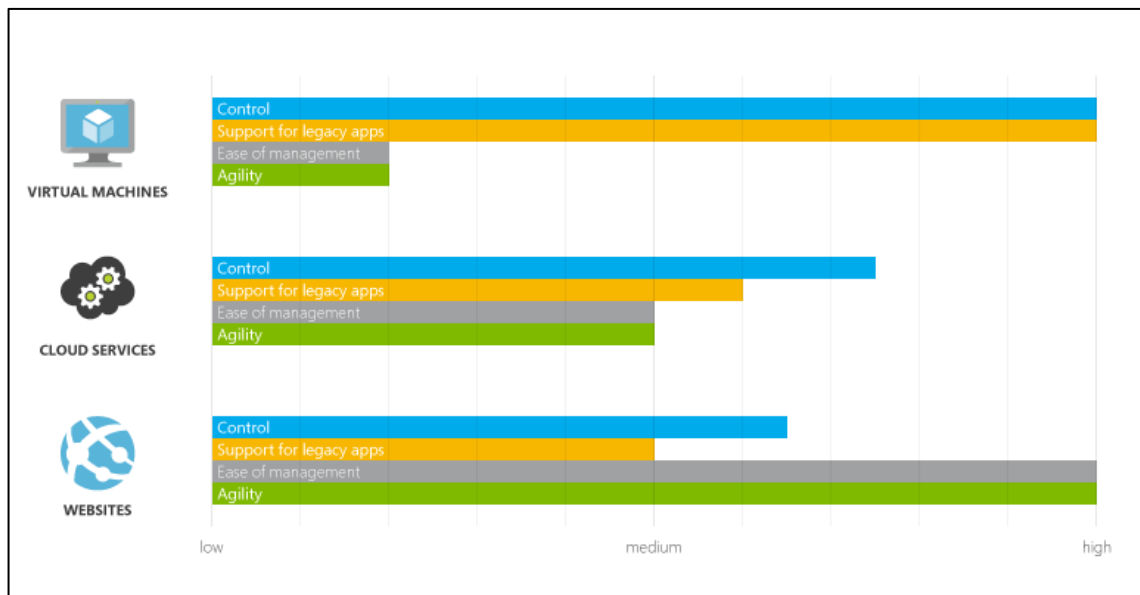
**Quadro 2: Opções de computação do Azure**

OPÇÕES DE COMPUTAÇÃO	PÚBLICO-ALVO
Serviço de Aplicativo	Aplicativos <i>Web</i> escalonáveis, Aplicativos Móveis, Aplicativos de API e Aplicativos Lógicos para qualquer dispositivo
Serviços de Nuvem	Aplicativos de nuvem de n camadas altamente disponíveis e escalonáveis com mais controle do sistema operacional
Máquinas virtuais	VMs Windows e Linux personalizadas com controle completo do sistema operacional

Fonte: Microsoft (2015).

O Serviço de Aplicativo do Azure é a melhor opção para a maioria dos aplicativos *Web*. A implantação e o gerenciamento estão integrados na plataforma, os sites podem ser dimensionados rapidamente para suportar altas cargas de tráfego e o gerenciador de balanceamento de carga e tráfego integrado oferece alta disponibilidade. É possível mover sites existentes para o serviço facilmente com uma ferramenta de migração online, usar um aplicativo de software livre da Galeria de Aplicativos Web ou criar um novo site usando a estrutura e as ferramentas de escolha do usuário. (DYKSTRA, 2015).

**Figura 10: o grau relativo de controle em relação à facilidade de uso de cada uma das opções de hospedagem na Web no Azure**



Fonte: Microsoft (2015).

O Serviço de Aplicativo permite que o usuário crie os seguintes tipos de aplicativos (KHAN, 2015):

- aplicativos *web* – É possível criar e implantar rapidamente aplicativos *Web* críticos dimensionados de acordo com o ambiente empresarial do cliente;
- aplicativos móveis – Permite criar aplicativos nativos e multiplataforma, conectar-se a sistemas corporativos, conectar-se facilmente a APIs SaaS, criar aplicativos prontos para uso *offline* com sincronização e criar notificações instantâneas de envio por *push* em qualquer dispositivo para milhões em segundos;
- aplicativos de API - permite compilar e consumir APIs em nuvem. Oferecem recursos para desenvolver, implantar, publicar, consumir e gerenciar APIs da Web RESTful;
- aplicativos lógicos - permitem que os desenvolvedores projetem fluxos de trabalho que comecem com um gatilho e então executem uma série de etapas, cada uma chamando um aplicativo de API do Serviço de Aplicativo ao mesmo tempo em que cuida com segurança da autenticação e práticas recomendadas, como definição de ponto de verificação e execução durável.

Se o usuário precisar de mais controle sobre o ambiente do servidor da *Web*, como a capacidade de acessar remotamente seu servidor ou configurar as tarefas de inicialização do

servidor, os Serviços de Nuvem do Azure geralmente são a melhor opção (DYKSTRA, 2015).

Os Serviços de Nuvem é um exemplo de PaaS (Plataforma como Serviço). Como os Serviços de Aplicativos, essa tecnologia foi desenvolvida para oferecer suporte a aplicativos escalonáveis, confiáveis e baratos de operar. São hospedados em VMs, no entanto, o usuário tem mais controle sobre as VMs do que nos Serviços de Aplicativos e pode instalar seu próprio software nas VMs do Serviço de Nuvem e controlá-los remotamente (GEORGE, 2015).

A tecnologia fornece duas opções de VM ligeiramente diferentes: as instâncias das funções *Web* executam uma variante do *Windows Server* com IIS, enquanto as instâncias das funções de trabalho executam a mesma variante do *Windows Server* sem IIS. Um aplicativo de Serviços de Nuvem depende de uma combinação dessas duas opções (GEORGE, 2015).

É importante notar que com os Serviços de Nuvem, o usuário não cria máquinas virtuais, apenas escolhe o tamanho que terão as VMs. É fornecido um arquivo de configuração que informa ao Azure quantas delas o usuário deseja e a plataforma as cria. Se o aplicativo precisar manipular uma carga de trabalho maior, é possível solicitar mais VMs, e o Azure criará essas instâncias. Se a carga diminuir, essas instâncias podem ser desligadas e elas não serão mais cobradas (GEORGE, 2015).

Caso a necessidade seja de um aplicativo existente que necessita de maiores modificações para ser executado no Serviço de Aplicativo do Azure ou nos Serviços de nuvem do Azure, é possível escolher as Máquinas Virtuais do Azure para simplificar a migração para a nuvem. No entanto, configurar, proteger e manter as máquinas virtuais corretamente requer muito tempo e conhecimento de TI em comparação ao Serviço de Aplicativo e aos Serviços de Nuvem. (MICROSOFT, 2015).

Através da plataforma Azure, a Microsoft oferece Infraestrutura como um Serviço (IaaS) com a opção de computação Máquinas Virtuais do Azure. Esse serviço permite criar e usar máquinas virtuais na nuvem possibilitando usar essa tecnologia de várias maneiras. Alguns exemplos incluem (MICROSOFT, 2015):

- a) Máquinas Virtuais (VMs) para desenvolvimento e teste - Usar VMs oferece uma maneira rápida e fácil de criar um computador com configurações específicas necessárias para testar o código e um aplicativo. As Máquinas Virtuais do Azure fornecem a facilidade de criar essas VMs, usá-las e removê-las quando elas não forem mais necessárias;



- b) recuperação de desastre - A recuperação de desastres com base na IaaS permite pagar pelos recursos de computação necessários somente quando precisar deles, não necessitando manter continuamente um datacenter de backup que raramente é usado;
- c) extensão do próprio datacenter para a nuvem pública - Usando a rede Virtual do Azure, uma organização pode criar uma rede virtual (VNET) - que é uma extensão da própria rede local - e adicionar VMs a esta VNET. Isso permite executar aplicativos como SharePoint, SQL Server e outros em uma VM do Azure. Essa abordagem pode ser mais fácil e barata de implantar do que executá-los em VMs do datacenter da própria organização.

Em uma VM no Azure o usuário pode usar uma imagem fornecida pelo Azure ou um de seus parceiros ou usar a sua própria. Os exemplos de sistemas possíveis incluem várias versões, edições e configurações de *Windows Server*, Servidores Linux (como Suse, Ubuntu e CentOS), *SQL Server*, *BizTalk Server* e *SharePoint Server* (MICROSOFT, 2015).

As máquinas virtuais usam discos rígidos virtuais (VHDs) para armazenar seus dados e sistema operacional (SO) e podem também ser usados para as imagens que o cliente optar por instalar um sistema operacional (DAVIES, 2015).

O sistema Azure faz o monitoramento do hardware físico que hospeda cada VM em execução, para que, se um servidor físico executando uma VM falhar, a VM seja movida para novo hardware e reinicializada. Azure também protege os dados de uma máquina virtual, mantendo cópias redundantes dos VHDs no armazenamento de blob (MICROSOFT, 2015).

O preço do serviço de Máquinas Virtuais do Azure é cobrado por hora com base no tamanho da VM e do sistema operacional. Para horas parciais, o Azure cobrará somente os minutos de uso. O armazenamento será cobrado separadamente (DAVIES, 2015).

O Azure possibilita a execução de aplicativos de alto desempenho ou HPC (*High-performance computing*) usando instâncias de computação de alto desempenho A8 e A9 e aproveitando uma rede de back-end com latência de MPI inferior a 3 microssegundos e taxa de transferência sem bloqueio de 32 Gbps. Esta rede de *back-end* inclui tecnologia RDMA (Acesso Remoto Direto à Memória) que habilita expandir aplicativos paralelos para milhares de núcleos. O Azure oferece alta memória CPUs de classe HPC para ajudar a obter resultados rápidos que podem expandir e reduzir de acordo a necessidade do cliente (MICROSOFT, 2015).

O RemoteApp, mais um dos serviços da Microsoft Azure, oferece a funcionalidade de

suporte pelos serviços de área de trabalho remota para o Azure, podendo usar nas plataformas Windows, Mac OS X, iOS ou Android. Os aplicativos da empresa cliente são executados no Windows Server na nuvem do Azure, onde é mais fácil expandir e atualizá-los. Os funcionários instalam clientes da Área de Trabalho Remota em seus *laptops*, *tablets* ou telefones conectados à Internet e podem, dessa maneira, acessar os aplicativos como se estivessem em execução localmente (POGGEMEYER, 2015).

A Microsoft Azure oferece duas opções para hospedar o SQL *Server*: Banco de Dados SQL do Azure e SQL Server na Máquina Virtual do Azure. Cada oferta pode ser caracterizada pelo nível de administração que o usuário tem sobre a infraestrutura e pelo grau de economia obtido com a consolidação de nível de banco de dados e automação (MICROSOFT, 2015).

O Banco de Dados SQL dá suporte para várias plataformas e tecnologias populares, como .NET, Java, PHP, Ruby on Rails e Node.js e oferece uma série de ferramentas de gerenciamento – APIs REST, PowerShell, o portal do Azure com suporte para HTML5 ou o SQL Server Management Studio. Com a integração do Visual Studio, o Azure oferece o desenvolvimento direto, *online* ou *offline*, em aplicativos locais ou criados para a nuvem. (TURKARSLAN, 2015).

O Banco de Dados de Documentos ou DocumentDB do Azure é um serviço de banco de dados de documento NoSQL projetado para suportar, de forma nativa, JSON e JavaScript diretamente dentro do mecanismo de banco de dados. Trata-se de uma boa solução para aplicativos que são executados na nuvem quando taxa de produtividade, baixa latência e consultas flexíveis são fundamentais (GENTZ, 2015).

Através de uma conta de armazenamento padrão, o usuário tem acesso a armazenamento dos seguintes tipos (MICROSOFT, 2015):

- a) o Armazenamento de Blob armazena dados de arquivos, não há restrições nos tipos de arquivos que pode ser carregado. Um blob pode ser qualquer tipo de texto ou dados binários;
- b) o Armazenamento de Tabela armazena conjuntos de dados estruturados. O Armazenamento de Tabela é um repositório de dados de atributo de chave NoSQL, que dá acesso rápido a grandes quantidades de dados;
- c) o Armazenamento de Fila fornece sistema de mensagens confiável para processamento de fluxo de trabalho e para comunicação entre componentes dos serviços de nuvem. É utilizado para ser criado um fluxo de trabalho escalonável e menos rígido entre diferentes funções do aplicativo;

- d) o Armazenamento de arquivos, usando o protocolo padrão SMB 2.1, oferece armazenamento compartilhado para aplicativos herdados.

Os dados no Armazenamento Azure também estão acessíveis via API REST, que pode ser chamado por qualquer linguagem que faça solicitações HTTP/HTTPS. O Armazenamento do Azure inclui garantias de consistência sólidas, além de permitir experiências preditivas para aplicativos criados no Azure (MICROSOFT, 2015).

A Microsoft oferece também uma solução de armazenamento em nuvem híbrida chamada Azure StorSimple. Ele usa um dispositivo proprietário (o dispositivo Microsoft Azure StorSimple), integração com serviços de nuvem e um conjunto de ferramentas de gerenciamento para proporcionar uma visualização de todo o armazenamento de empresa, incluindo armazenamento em nuvem, e oferece também escalonamento maciço de infraestrutura de armazenamento, ao usar o Azure para armazenar dados primários inativos de rápido crescimento (SMITH, 2015).

A Microsoft projeta a segurança no código de software seguindo uma abordagem conhecida como *Security Development Lifecycle* (SDL). Este processo de desenvolvimento insere requisitos de segurança em todo o ciclo de vida do software, desde o planejamento até a implantação (MICROSOFT, 2015).

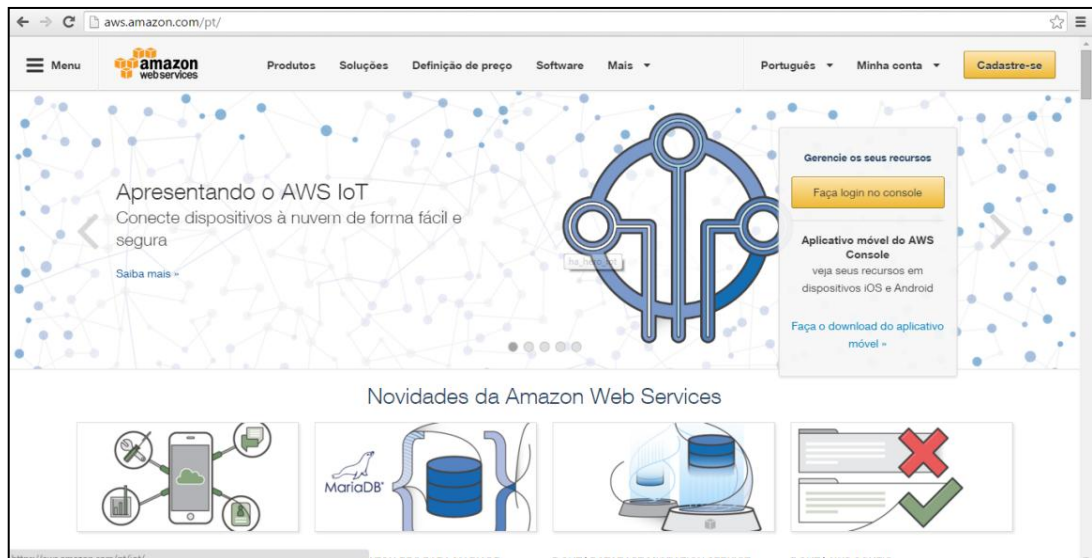
A plataforma Microsoft Azure ainda oferece o Microsoft Antimalware para a proteção contra ameaças online dos serviços de nuvem e máquinas virtuais. A Microsoft também emprega prevenção de ataques DDoS (ataque de negação de serviço distribuído), detecção de intrusão, testes de penetração regulares e ferramentas de análises de dados e aprendizado de máquina para ajudar a diminuir consideravelmente as ameaças contra a plataforma Azure. A Rede Virtual do Azure estende-se à sua rede local para a nuvem por meio de VPN site a site (MICROSOFT, 2015).

#### **2.5.4 Amazon Web Services (AWS)**

Amazon Web Services (AWS) é um conjunto de serviços de computação remota (também chamados *web services*) que juntos, constituem uma plataforma de computação em nuvem, proporcionada através da Internet pela Amazon.com (ver Figura 11) (AMAZON, 2015). A AWS oferece um conjunto amplo de serviços globais de computação,

armazenamento, banco de dados, análise, aplicativos e implementação e apresenta características relacionadas à computação em nuvem como alto nível de desempenho, eficácia, escalabilidade, disponibilidade, elasticidade e economia.

**Figura 11: Página inicial do site da AWS Amazon.**



Fonte: Print Screen do site no navegador Google Chrome.

Uma das pioneiras na tecnologia de computação em nuvem, a Amazon começou a oferecer serviços que possibilitavam utilizar os recursos computacionais de seus Datacenters em 2006. Isso aconteceu devido ao fato da Amazon ter adquirido uma infraestrutura muito poderosa para garantir que a demanda gerada em períodos de picos de vendas do seu site seria atendida. Nos momentos de períodos de baixa demanda, muitos recursos eram inativos; por isso a Amazon optou por vender a capacidade computacional excedente (Guzmán, 2011).

A AWS conta com uma enorme variedade de serviços em nuvem (mais de 500 atualmente). Para este estudo iremos explicar sobre os serviços EC2, *Elastic Beanstalk*, S3 e RDS.

#### 2.5.4.1 Amazon *Elastic Compute Cloud* (EC2)

O Amazon *Elastic Compute Cloud* (Amazon EC2) é um serviço da web que fornece capacidade computacional na nuvem e que permite os usuários utilizarem máquinas virtuais nas quais podem executar suas próprias aplicações fornecendo um controle completo sobre

seus recursos computacionais e permitindo a execução no ambiente computacional (LECHETA, 2014).

As aplicações dos usuários são configuradas através de uma imagem Amazon *Machine Image* (imagem de máquina de Amazon) também conhecida como instância (GUZMÁN, 2011). O usuário poderá usar um modelo de Imagem de Máquina da Amazon (AMI) pré-configurada, para começar a usar o serviço imediatamente, ou criar uma AMI contendo as próprias aplicações, bibliotecas, dados e definições de configuração associadas (AMAZON, 2015).

O Amazon EC2 fornece uma série de recursos para criar aplicativos escaláveis e resistentes a falhas. Entre eles estão (AMAZON, 2015):

- a) *Amazon Elastic Block Store* - Oferece armazenamento persistente para as instâncias do Amazon EC2. Permite criar volumes de armazenamento e anexá-los a instâncias. Os dados dos volumes do Amazon EBS são replicados em vários servidores em uma zona de disponibilidade para evitar perdas de dados causadas por falha em qualquer componente único. Os volumes do Amazon EBS são vinculados à rede e persistem independentemente da vigência de uma instância.
- b) *Amazon Virtual Private Cloud* – A *Amazon Virtual Private Cloud* (Amazon VPC) permite provisionar uma seção da nuvem da *Amazon Web Services* (AWS) isolada logicamente onde o cliente pode executar recursos da AWS em uma rede virtual que ele mesmo defina. O usuário tem controle total sobre seu ambiente de rede virtual, incluindo a seleção do seu próprio intervalo de endereços IP, criação de subnets e configuração de tabelas de roteamento e *gateways* de rede. Além disso, a Amazon VPC possibilita criar uma conexão de *Hardware Virtual Private Network* (VPN) entre um datacenter corporativo e a VPC e aproveitar a nuvem da AWS como uma extensão do datacenter da corporação cliente.
- c) *Endereços Elastic IP* - Os endereços Elastic IP são endereços de IP estáticos projetados para computação em nuvem dinâmica. Um endereço Elastic IP não está associado a uma instância específica, mas sim a uma conta, possibilitando ao cliente controlar esse endereço até que decida explicitamente liberá-lo.
- d) *Amazon CloudWatch* - Web service que permite aos clientes monitorarem em tempo real todos os recursos da instância como hardware e rede.
- e) *Cluster de Computação de Alta Performance (HPC)* – Serviço que possibilita os clientes alcançarem o mesmo alto desempenho computacional e de rede fornecido por

uma infraestrutura personalizada, enquanto se beneficiam das vantagens de flexibilidade, elasticidade e de custo do Amazon EC2. Esse serviço é indicado para clientes com complexas cargas de trabalho computacionais, assim como processos paralelos fortemente acoplados ou com aplicativos sensíveis ao desempenho da rede;

f) Instâncias de GPU – A AWS disponibiliza ainda o serviço de instancias de GPU com alta performance paralela e que oferecem acesso às GPUs NVIDIA, cada uma com até 1.536 núcleos CUDA e 4 GB de memória de vídeo. Com as versões mais atuais do driver, esses GPUs são compatíveis com OpenGL, DirectX, CUDA, OpenCL e GRID SDK;

g) Instâncias de E/S elevada - Os clientes que exigem acesso de E/S aleatória, de baixa latência e muito elevada a seus dados podem se beneficiar de instâncias de E/S elevada. As instâncias de E/S elevada são um tipo de instância do Amazon EC2 que pode fornecer aos clientes taxas de E/S aleatória em 100.000 IOPS;

h) Instâncias de armazenamento denso - As instâncias com armazenamento denso são um tipo de instância do Amazon EC2 que pode fornecer aos clientes taxa de transferência de E/S sequencial de 3,9 GB/s e 48 TB de armazenamento de instância em 24 discos rígidos.

Como o próprio nome já diz, o Amazon *Elastic Compute Cloud* é altamente escalável. Este permite que o cliente aumente ou diminua a capacidade em minutos. É possível comissionar uma, centenas ou até milhares de instâncias do servidor simultaneamente. Naturalmente, como tudo é controlado com os serviços de APIs da web, o aplicativo pode automaticamente se expandir ou se reduzir, dependendo da necessidade.

O usuário pode escolher tipos de várias instâncias, sistemas operacionais e pacotes de software. O Amazon EC2 permite que seja selecionada uma configuração de memória, CPU, armazenamento de instância e tamanho da partição de inicialização que seja ideal para a opção de sistema operacional e aplicativos. O EC2 disponibiliza atualmente os sistemas operacionais Amazon Linux, Windows *Server*, CentOS e Debian (GNU/Linux).

O serviço é executado dentro dos datacenters e da infraestrutura de rede da Amazon. Para garantir uma menor latência, A AWS oferece 11 regiões localizadas em todo o mundo para localização dos dados e 28 zonas de disponibilidade redundantes. O compromisso do Acordo de Nível de Serviço do Amazon EC2 é disponibilidade de 99,95% para cada região (AMAZON, 2015).

O Amazon EC2 trabalha em conjunto com a Amazon VPC para oferecer funcionalidades de rede seguras e robustas para os seus recursos de computação. Uma dessas funcionalidades é a possibilidade de provisionar os recursos de EC2 como instâncias dedicadas. Instâncias dedicadas são instâncias do Amazon EC2 que são executadas em hardware dedicado a um único cliente para proporcionar um isolamento adicional (LECHETA, 2015).

Amazon EC2 é que permite escolher entre 3 tipos de pagamentos. A escolha do usuário depende da capacidade computacional que irá realmente utilizar (AMAZON, 2015):

- a) instâncias sob demanda – As instâncias sob demanda permitem que você pague pela capacidade de computação por hora sem compromissos de longo prazo;
- b) instâncias reservadas – Com essa opção o usuário pode reservar uma instância (fazendo um único pagamento reduzido para cada instância que se deseja reservar), podendo optar pela execução da instância pela taxa horária reduzida no seu prazo (desconto significativo durante uma taxa horária) ou pode optar por não pagar taxa de utilização (enquanto não estiver em uso a instância);
- c) instância *Spot* – Boa para clientes que possuem flexibilidade sobre quando os seus aplicativos podem ser executados, as instâncias *Spot* permitem aos clientes negociarem a capacidade não utilizada do Amazon EC2 e executarem essas instâncias durante o período em que sua oferta exceder o preço *spot* atual.

#### 2.5.4.2 AWS *Elastic* Beanstalk

O AWS *Elastic* Beanstalk é um serviço para implantação, escalabilidade e execução de aplicações e serviços da web no ambiente computacional da Amazon. A plataforma possibilita o desenvolvimento de aplicações com uma vasta lista de linguagens de programação em servidores familiares como Apache e IIS (AMAZON, 2015).

Com o *Elastic* Beanstalk, os desenvolvedores possuem o controle total sobre os recursos da AWS e podem desempenhar uma série de funções ao ajustar definições de configuração padrão do console de gestão do serviço. Além disso, após o cliente fazer o *upload* do seu código, o *Elastic* Beanstalk se encarrega automaticamente do provisionamento de capacidade, do balanceamento de carga, da escalabilidade automática e até o

monitoramento da saúde do aplicativo. Não é cobrado adicional algum pelo *Elastic Beanstalk*, o usuário paga apenas pelos recursos da AWS necessários para executar e armazenar seus aplicativos. Além disso, é oferecido ao cliente doze meses de acesso ao nível de uso gratuito da AWS (LECHETA, 2014).

#### 2.5.4.3 Amazon *Simple Storage Service* (Amazon S3)

O Amazon S3 é um serviço de armazenamento por demanda que apresenta uma interface de serviço simples para armazenar e recuperar qualquer quantidade de dados de qualquer parte da Web. Não é cobrada taxa mínima e nem custo de configuração e quando há necessidade de mais ou menos espaço de armazenamento, o sistema automaticamente aumenta ou reduz o espaço alocado. O nível gratuito da AWS inclui 5 GB de armazenamento, 20.000 requisições *GET* e 2.000 requisições *PUT* para o Amazon S3 (AMAZON, 2015).

O Amazon *Simple Storage Service* (Amazon S3) armazena dados como objetos dentro de recursos denominados "*buckets*". O *bucket* permite armazenar, gravar, ler e excluir objetos que podem ter até 5 terabytes de tamanho. Esses recursos podem ser controlados pelo usuário, possibilitando a visualização de logs de acesso e ainda a escolha da região da AWS onde o *bucket* será armazenado (LECHETA, 2014).

O serviço S3 oferece opções de categorias de armazenamento que devem ser escolhidas de acordo com o uso, como:

- a) o Amazon S3 Standard para armazenamento com aplicações variadas de dados acessados;
- b) o Amazon S3 Standard – Acesso ocasional (Standard – IA) para dados de longa duração, porém acessados com menos frequência;
- c) Amazon Glacier para arquivos de longa duração.

#### 2.5.4.4 Amazon *Relational Database Service* (Amazon RDS)

O Amazon *Relational Database Service* (RDS) é um serviço para o uso de bancos de dados relacionais na nuvem da AWS. O Amazon RDS facilita a configuração, a operação e a



escalabilidade de bancos de dados, ajudando na execução de tarefas rotineiras de banco de dados como provisionamento, aplicação de patches, backup, recuperação, detecção de falhas e reparo, além de fornecer uma capacidade econômica e redimensionável. O cliente paga somente pela capacidade de computação por hora de execução da instância de banco de dados, sem nenhum compromisso de longo prazo (AMAZON, 2015).

O Amazon RDS oferece acesso aos recursos dos sistemas de gerenciamento dos bancos de dados relacional Amazon Aurora, MySQL, Oracle, Microsoft SQL Server ou PostgreSQL. O usuário poderá acessar os recursos através do Console de Gerenciamento da AWS, da interface da linha de comando do Amazon RDS ou de chamadas de API simples (LECHETA, 2014).

#### 2.5.4.5 Amazon DynamoDB

O Amazon DynamoDB é um serviço de banco de dados na nuvem NoSQL indicado para todos os aplicativos que precisam de latência consistente abaixo de 10 milissegundos em qualquer escala. O serviço é totalmente gerenciado – o usuário precisa apenas criar uma tabela de banco de dados e definir a taxa de transferência – e é compatível com os modelos de armazenamento de documentos e de chave-valor. Seu modelo de dados flexível e desempenho confiável fazem dele a escolha indicada para aplicativos da Web e móveis, jogos, tecnologia de anúncios, IoT, etc (AMAZON, 2015).

O banco de dados DynamoDB, com o uso do AWS SDK, permite escrever aplicativos que armazenam documentos JSON diretamente nas tabelas, reduzindo a necessidade de criação de código para inserir, atualizar e recuperar documentos JSON e permitindo executar operações poderosas usando apenas algumas linhas de código como consultas JSON aninhadas (AMAZON,2015).

O Amazon DynamoDB oferece ainda suporte ao armazenamento, à consulta e à atualização de documentos e é compatível com estruturas de dados de chave-valor. Nessa estrutura cada item (linha) é um par chave-valor onde a chave primária identifica unicamente cada um e é o único atributo obrigatório dos itens de uma tabela. Cada item pode ter qualquer número de atributos (colunas) (AMAZON,2015).

## 2.6 TRABALHOS RELACIONADOS

Em Gomes (2012) foi feito um estudo de caráter bibliográfico sobre computação em nuvem e suas principais plataformas. Também se comparou duas plataformas de mercado em nível de funcionalidades, o *App Engine* da Google e o Windows Azure da Microsoft. A autora apresenta que a computação em nuvem é o futuro e presume que grandes empresas irão migrar para essa tecnologia.

Gomes (2012) apresenta no seu estudo comparativo entre *App Engine* e Windows Azure, que ambas possuem semelhanças e estão no mesmo patamar de utilidade e funcionalidade. Entretanto, as duas têm algumas características distintas e que melhor se adequam a determinados grupos de clientes.

O artigo Dias e Oliveira Junior (2012) mostra como configurar, migrar, implantar e executar uma aplicação de Gerenciamento de Protocolos Java EE 6 no serviço OpenShift. Inicialmente, os autores apresentam os conceitos principais da computação em nuvem, o funcionamento da plataforma OpenShift e as tecnologias que foram utilizadas para o processo de implantação da aplicação “Sistemas de Protocolos”. Em seguida, o artigo apresenta os passos da configuração do OpenShift até a execução na plataforma. É apresentada ainda toda a descrição da aplicação utilizada.

Dias e Oliveira Junior (2012) explicam que tiveram que fazer algumas alterações nos arquivos para a execução na plataforma e concluem que o OpenShift oferece disponibilidade, segurança e confiabilidade e vem se mostrando como uma excelente alternativa para os desenvolvedores de software que pretendem iniciar os seus experimentos em computação em nuvem.

No trabalho de Aquino (2013) é apresentada a plataforma *App Engine* da Google, mostrando os passos básicos para o desenvolvimento de uma aplicação simples em Java, utilizando a IDE de desenvolvimento Eclipse. Além disso, é apresentada uma comparação com os recursos do AWS *Elastic Beanstalk* da Amazon. O autor explica que seguiu os passos propostos pela própria documentação oficial do Google *App Engine*, e funcionou de acordo com o propósito apresentado no trabalho. Ele conclui que o serviço *Elastic Beanstalk* da Amazon apresenta melhores resultados na utilização de APIs Java e que o projeto no *App Engine* mostrou bons resultados, mostrando que a plataforma presta um serviço confiável.

### 3 ESTUDO COMPARATIVO ENTRE OPENSIFT E ELASTIC BEANSTALK AWS

Neste capítulo será apresentado um estudo comparativo entre os serviços prestados pela plataforma Openshift da empresa *Red Hat*, na modalidade *Online*, e o serviço *Elastic Beanstalk* do AWS da empresa Amazon. Estes foram escolhidos para esse estudo por ambos fornecerem serviços em nuvem no modelo de serviço PaaS, apresentarem facilidades para implantação de aplicações e estarem entre os principais serviços em nuvem do mercado. Para alcançar o objetivo foram comparados pontos como: quais linguagens de programação e servidores de aplicação são suportados, o que é oferecido como serviço gratuito, quais regiões dos *datacenters* são oferecidos para a modalidade gratuita, quais Bancos de Dados são oferecidos e como cada um trata o controle de versão.

Foi usado também como método de estudo a implantação de uma aplicação usando banco de dados nas plataformas. Como meio de criação e configuração dos ambientes e da aplicação foram usados o Console de cada serviço e um software para administração do banco de dados.

Para demonstrar a implantação de aplicativos nas duas plataformas, foi usada uma aplicação *Open Source* de controle financeiro, desenvolvida pelo programador identificado como Tales Henrique e suas principais funcionalidades podem ser vistas no diagrama da Figura 12. O projeto pode ser encontrado e feito o *download* no repositório do autor no serviço de hospedagem web GitHub (GITHUB, 2015). A aplicação foi desenvolvida na linguagem PHP e utiliza o MySQL como banco de dados. Para a interface, foi usado o *framework front-end* Bootstrap. Para administração do banco de dados foram usados o aplicativo web phpMyAdmin para o Openshift e a ferramenta MySQL *Workbench* com o *Elastic Beanstalk*.

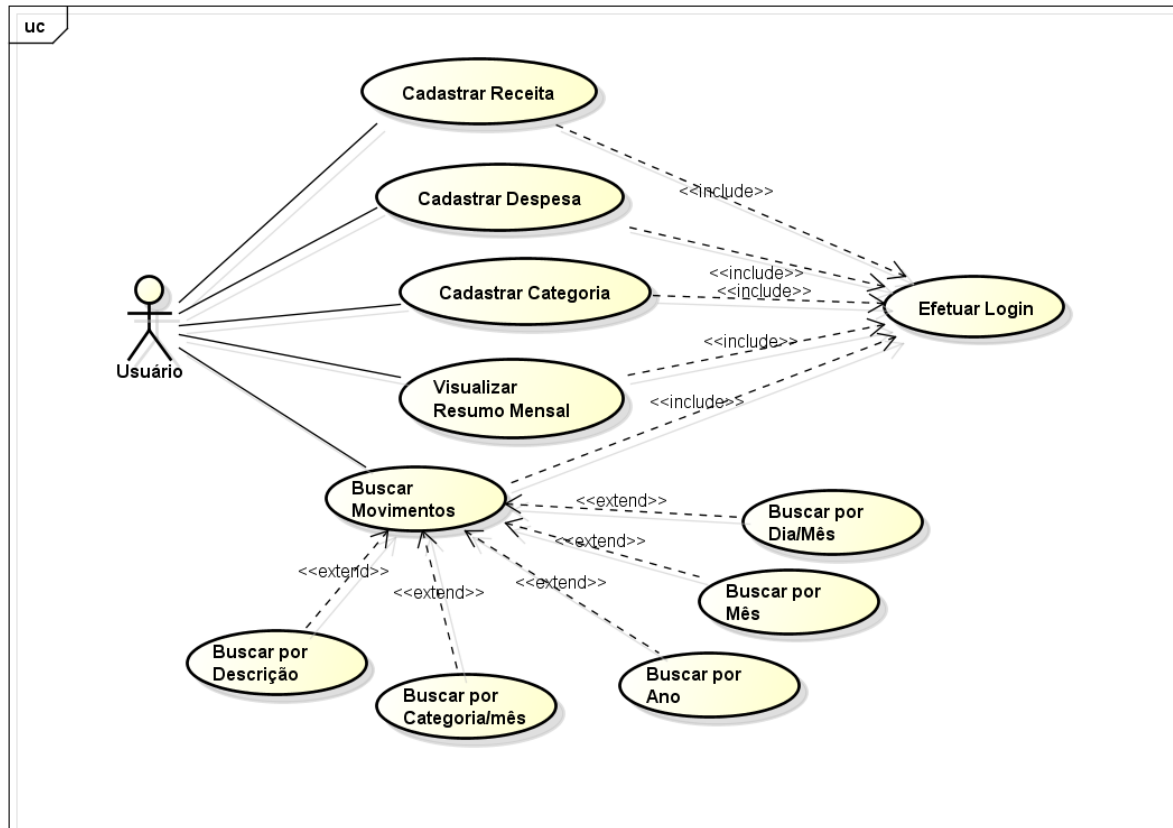
Também Foi realizada uma medição da velocidade de carregamento da página “index.php” da aplicação nas duas plataformas. Para isso, foi utilizada a ferramenta Firebug v2.0.13 em formato de *plugin* para o navegador Mozilla Firefox. O Firebug (FIREBUG, 2015) está disponível gratuitamente e dispõe de diversas ferramentas, como monitor de HTML, CSS, JavaScript e tempo de carregamento dos objetos que compõem a página. Estas ferramentas geralmente são utilizadas por desenvolvedores para identificar e eliminar erros de edição e programação das páginas web.

Para testar o desempenho das plataformas com várias conexões, foi usada a ferramenta Httpperf. Esta é executada na máquina cliente e tem como propósito medir a performance de

um servidor *web*.

Ainda para a realização dos testes, vale salientar que foi utilizado um *notebook* com processador i3e 4GB de memória, utilizando uma internet banda larga de 5 Mbps.

Figura 12: Diagrama de Caso de uso da aplicação Controle Financeiro.



Fonte: Produção do próprio autor.

### 3.1 CARACTERÍSTICAS DO OPENSIFT

A plataforma OpenShift *Online* da *Red Hat* se baseia no modelo de serviço PaaS, oferecendo ao usuário uma plataforma desenvolvimento sem ter que se preocupar com a infraestrutura do servidor da aplicação. Nessa plataforma é possível criar aplicações auto-escaláveis.

No Openshift as aplicações rodam em máquinas virtuais chamadas *gears*, compostas por serviços chamados de cartuchos (*cartridges*). Por padrão, as aplicações rodam em um sistema operacional *Red Hat Linux*, sendo que no plano gratuito do OpenShift *Online*, é oferecida ao usuário a possibilidade de criar até três *small gears*, cada uma provendo 512 MB

de RAM, 100 MB de espaço *swap* e 1GB de armazenamento. É importante salientar que o usuário pode mudar seu plano caso aja necessidade de mais recurso, pagando de acordo com o uso. Essa mudança acontece por desejo explícito do cliente.

### 3.1.1 Linguagens e Servidores de Aplicação Compatíveis

O serviço OpenShift *online* suporta uma grande variedade de linguagens de programação, sendo elas Ceylon, Go Language, Hack, Java, Node.js, Perl, PHP, Python, Ruby e Scala. É possível ainda trabalhar com muitos *Frameworks* e ferramentas de desenvolvimento integradas, como a integração Eclipse, JBoss Developer Studio e Jenkins. Como servidores de aplicação, segundo a documentação do Openshift (OPENSIFT b, 2015), a plataforma suporta JBoss AS, JBoss EAP, WildFly, Tomcat, Zend Server e Vert.x.

### 3.1.2 Suporte a Banco de Dados

No Openshift o banco dados para a aplicação é criado anexando o cartucho adequado. Esse cartucho pode ser criado no console - apenas clicando na opção - ou através da ferramenta RHC, onde deve ser informado o cartucho a ser anexado ou *link* de *download*. No código fonte da aplicação, para a conexão com o BD, deve-se colocar as variáveis de ambiente do BD usado. A plataforma oferece opção aos bancos dados MySQL, MongoDB, PostgreSQL, SQLite e Amazon RDS. Alguns serviços para administração do Banco de Dados são oferecidos como cartuchos, facilitando a instalação.

### 3.1.3 Segurança

A segurança sobre a infraestrutura é a mesma para ambos os serviços, já que o Openshift é executado usando instâncias EC2 da Amazon. Estas oferecem vários recursos de segurança como: solicitações de API só por assinatura, uso de criptografia no serviço S3 usando SSL ou

criptografia *client-side*, Credenciais de segurança da AWS, etc.

Por ter como base o *Red Hat Enterprise Linux*, o OpenShift fornece um ambiente isolado seguro que incorpora os três seguintes mecanismos de segurança (RED HAT, 2015):

- a) SELinux é uma implementação de um mecanismo de controle de acesso obrigatório (MAC) no *kernel* do Linux. Ele verifica se há operações permitidas em um nível além do que os controles de acesso discricionários padrão (DAC) fornecem. SELinux fornece um alto nível de isolamento entre os aplicativos em execução dentro do OpenShift porque cada *gear* e seu conteúdo são rotulados de forma exclusiva;
- b) Grupos de Controle permitem alocar processador, memória e recursos de entrada e saída (I/O) entre aplicações. Eles fornecem o controle de utilização de recursos em termos de consumo de memória, armazenamento, utilização de rede para entrada/saída e prioridade do processo. Isso permite a criação de políticas de alocação de recursos, garantindo assim que nenhum recurso do sistema consuma todo o sistema e afete outros serviços;
- c) Kernel *namespaces* separa grupos de processos para que eles não possam ver os recursos em outros grupos. Da perspectiva de um aplicativo OpenShift executando, por exemplo, o aplicativo tem acesso a um sistema *Red Hat Enterprise Linux* em execução, embora pudesse ser um dos muitos aplicativos em execução dentro de uma única instância do mesmo sistema.

### 3.1.4 Controle de Versão

O serviço OpenShift utiliza para o controle de versões a tecnologia *Git* para acessar um repositório. Dessa forma o usuário pode codificar localmente e em seguida, fazer um *push* de suas alterações para o servidor. O servidor, em seguida, executa uma série de ganchos para construir e configurar o aplicativo e, finalmente, reinicia o aplicativo.

### 3.1.5 Regiões Disponíveis Gratuitamente

Por usarem a mesma infraestrutura da Amazon, o Openshift e o *Elastic Beanstalk* podem lançar suas instâncias em qualquer das regiões da Amazon apresentadas no Quadro 3. Entretanto, usuários que possuem o plano gratuito do Openshift Online podem criar as *gears* somente na região us-east-1.

**Quadro 3: Regiões disponíveis na Amazon**

Code	Name
ap-northeast-1	Asia Pacific (Tokyo)
ap-southeast-1	Asia Pacific (Singapore)
ap-southeast-2	Asia Pacific (Sydney)
eu-central-1	EU (Frankfurt)
eu-west-1	EU (Ireland)
sa-east-1	South America (Sao Paulo)
us-east-1	US East (N. Virginia)
us-west-1	US West (N. California)
us-west-2	US West (Oregon)

Fonte: Amazon (2015)

### 3.2 CARACTERÍSTICAS DO AWS ELASTIC BEANSTALK

Na plataforma da Amazon, as máquinas virtuais são chamadas de instâncias. No plano gratuito do Beanstalk é oferecido, por um período de doze meses, 750 horas por mês de uso de instância t2.micro (Processadores Intel Xeon e até 1GB de Memória) com sistema operacional Amazon Linux AMI ou o Windows Server 2008 R2 AMI (podendo usar outros sistemas acessando diretamente a instância EC2), executando uma instância de aplicação por mês ou duas por duas semanas. Para armazenamento é disponibilizado 5 GB do produto Amazon S3. Para executar uma aplicação é criada uma instância EC2 e caso queira usar algum banco de dados, será criada uma instância do BD escolhido. Caso o usuário ou a

aplicação necessite de mais recursos, será cobrado automaticamente pelo uso excedido. Importante lembrar que as informações do Cartão de Crédito do cliente são pedidas no ato do cadastro.

### 3.2.1 Linguagens e Servidores de Aplicação Compatíveis

Através do Elastic Beanstalk, é possível criar uma aplicação padrão em PHP, Java, Python, Ruby, Node.js, .NET ou usar a tecnologia Docker. Quando é escolhida a linguagem, a própria plataforma já cria o servidor web e o servidor de aplicativos apropriado para a linguagem. Frameworks são instalados gerenciando diretamente a instância EC2 criada para a aplicação.

O AWS *Elastic* Beanstalk suporta as seguintes pilhas de software (LECHETA, 2014):

- a) aplicativos Apache Tomcat para Java;
- b) aplicativos Apache HTTP Server para PHP e Python;
- c) Nginx e Apache HTTP Server para aplicativos Node.js;
- d) *Passenger* para aplicativos Ruby;
- e) Microsoft IIS 7.5 para aplicativos em .NET.

### 3.2.2 Suporte a Banco de Dados

Na criação de aplicação facilitada do Beanstalk, é dada ao usuário a possibilidade de criação de uma instância do serviço Amazon RDS. Esse serviço disponibiliza a utilização dos seguintes mecanismos de banco de dados: Amazon Aurora, Oracle, Microsoft SQL *Server*, PostgreSQL, MySQL e MariaDB. Caso a necessidade seja de outro serviço de armazenamento, deve ser usada uma instância deste serviço usando a mesma rede de serviço. Serviços de administração de BDs podem ser instalados por linha de comando, usando aplicativos de conexão remota ou o usuário pode usar alguma ferramenta *desktop*.



### 3.2.3 Segurança

A segurança sobre a infraestrutura é a mesma para ambos os serviços, já que o Opsshift é executado usando instâncias EC2 da Amazon. Estas oferecem vários recursos de segurança como: solicitações de API só por assinatura, uso de criptografia no serviço S3 usando SSL ou criptografia *client-side*, Credenciais de segurança da AWS, etc.

A Amazon oferece outros recursos próprios para seus serviços - incluindo o AWS Elastic Beanstalk - para garantir segurança no gerenciamento das instâncias. Um desses serviços é a Amazon *Virtual Private Cloud* (Amazon VPC) que fornece a possibilidade de criar uma rede virtual entre as instâncias criadas ou isolar outras, tornando o gerenciamento das instâncias mais seguro. O cliente pode usar VPC para provisionar uma seção privada e isolada de seu aplicativo – que por padrão é público - em uma rede virtual que o usuário definir. Este serviço provê também a possibilidade do cliente poder integrar via VPN (*Virtual Private Network*), viabilizando a comunicação entre seu ambiente na Amazon e outro ambiente, tornando a nuvem em uma extensão do *datacenter* corporativo do usuário. O usuário poderá também usar um VPN de software terceirizado para criar uma conexão VPN de site a site ou de acesso remoto com a VPC por meio do *Gateway da Internet*.

O AWS oferece também o *Identity and Access Management* (IAM) que permite o cliente gerenciar os usuários e grupos da AWS e suas permissões para negar ou permitir o acesso a determinados recursos da plataforma.

### 3.2.4 Controle de Versão

Para o controle de versões do Elastic Beanstalk existem muitas opções de ferramentas incluindo o *AWS Management Console* – fazendo apenas um *upload* -, a implementação do Git e a interface da linha de comando, o *AWS Toolkit for Visual Studio* e o *AWS Toolkit for Eclipse*.

### 3.2.5 Regiões Disponíveis Gratuitamente

Saber as regiões de implantação é importante para que o usuário possa levar em consideração os custos, a latência de rede e os recursos suportados.

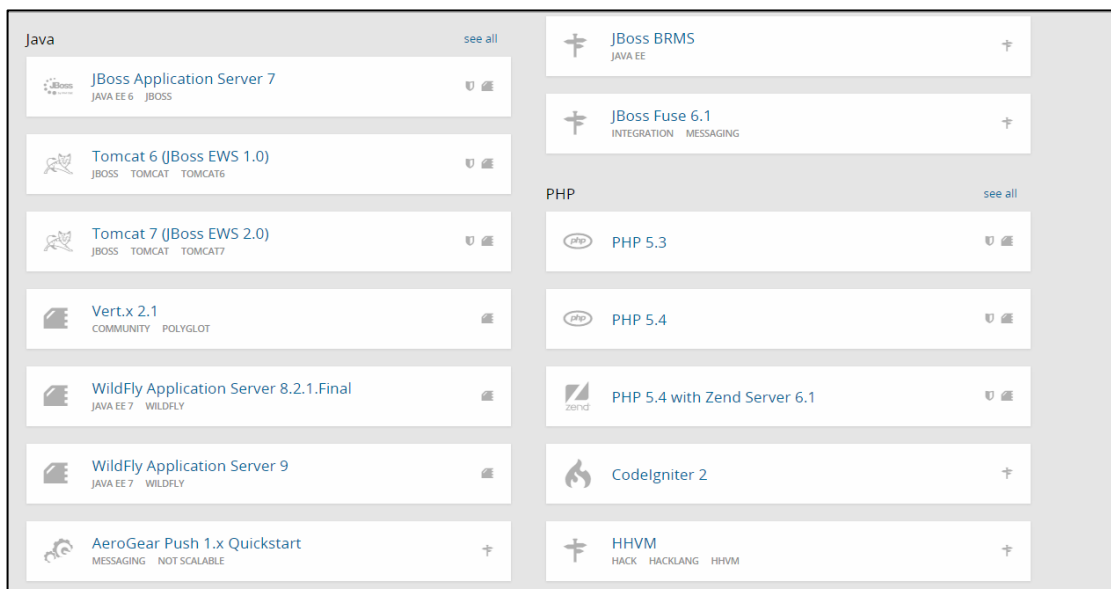
Os usuários da AWS, no período gratuito da Amazon, podem criar suas instâncias em qualquer das regiões descritas no Quadro 3.

### 3.3 CENÁRIOS E TESTES DE DESEMPENHO

#### 3.3.1 Implantação do Projeto no OpenShift

Com a conta do usuário criada no site da OpenShift, temos acesso ao Console da plataforma. No início da configuração de uma nova aplicação é dada ao usuário a opção de escolher o tipo de aplicação que será criada, sendo que, para este trabalho, foi escolhido o cartucho PHP 5.4. É possível perceber essa etapa na Figura 13.

**Figura 13: Escolhendo o tipo de aplicação.**



Fonte: Print screen do site no navegador Google Chrome.

No próximo passo, o serviço pede para que seja informado um nome de domínio. Esse domínio se torna um sufixo das URLs exclusivas para a conta Openshift criada. É preciso também informar o nome da aplicação que é parte fundamental para acesso através da URL.

Para esse projeto foi escolhido o domínio “tccdenys” e nome da aplicação “controlefinanceiro”, como é possível ver na Figura 14.

Ainda no mesmo passo de configuração, a plataforma apresenta a opção de colocar um endereço de repositório, caso o usuário deseje que a aplicação inicie já com uma cópia do projeto. Nessa configuração a plataforma ainda apresenta o tamanho da *Gear*, o cartucho escolhido e a opção de deixar a aplicação escalável de acordo com o tráfego web, opção escolhida para esta implantação. Depois é necessário clicar no botão “*Create Application*”, criando de fato a máquina virtual para a aplicação.

**Figura 14: Configurando a aplicação do Openshift.**

Based On **PHP 5.4 Cartridge**

PHP is a general-purpose server-side scripting language originally designed for Web development to produce dynamic Web pages. Popular development frameworks include CakePHP, Zend, Symfony, and Code Igniter.

<http://www.php.net>

- ☆ OpenShift maintained
- 🔒 Receives automatic security updates

Public URL

OpenShift will automatically register this domain name for your application. You can add your own domain name later.

Source Code

We'll create a Git code repository in the cloud, and populate it with a set of reasonable defaults. If you provide a Git URL, your application will start with an exact copy of the code and configuration provided in this Git repository.

Gears **small**

Gears are the application containers running your code. For most applications, the small gear size provides plenty of resources. You can also [upgrade your plan](#) to get access to more gear sizes.

Cartridges **PHP 5.4**

Applications are composed of cartridges - each of which exposes a service or capability to your code. All applications must have a web cartridge.

Scaling **No scaling**

OpenShift automatically routes web requests to your web gear. If you allow your application to scale, we'll set up a load balancer and allocate more gears to handle traffic as you need it.

Region  No preference  **aws-us-east-1**  aws-eu-west-1

All gear sizes can be deployed to the US Region.

WARNING: Small gears cannot be deployed to this region. Only production gears can be deployed to the EU Region (small,highcpu, medium)

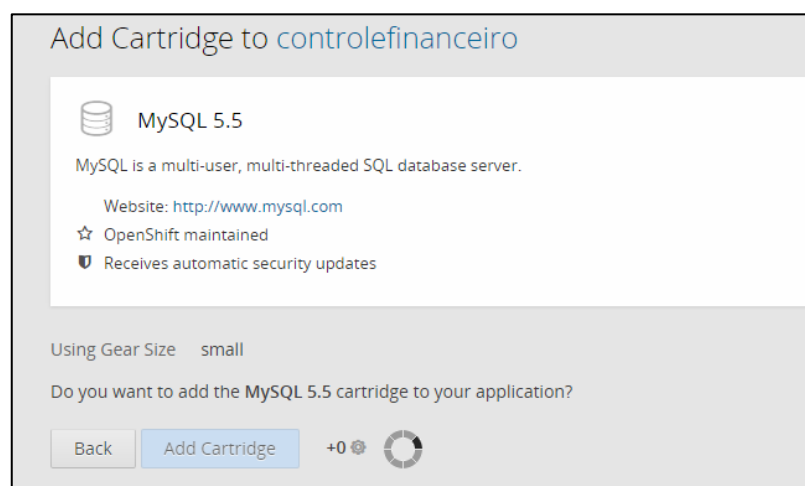
Fonte: Print screen do site no navegador Google Chrome.

Com a aplicação criada, o Openshift apresenta a URL da aplicação, o endereço do repositório – chamado de *Code Source* - e oferece a possibilidade de incluir outros cartuchos como serviço, ainda através do console web. Clicando no cartucho adequado, foi incluído o serviço MySQL 5.5 (Figura 15). Ao instalar este, a plataforma informa a url de acesso ao banco de dados, o usuário e senha que devem ser usados para a conexão. Nota-se que no endereço de acesso ao BD, o Openshift apresenta apenas as variáveis de ambiente, recurso que a plataforma utiliza caso haja alguma alteração no endereço de acesso do banco de dados.

Depois da instalação do MySQL na *Gear* gerada, foi escolhida também a opção

oferecida pela plataforma de instalar o cartucho phpMyAdmin 4.0. Com a adição desse sistema de administração de banco de dados, a plataforma informa a URL de acesso – que baseia no endereço da aplicação acrescido de /phpmyadmin – e informa o usuário a senha, que são os mesmos do MySQL.

**Figura 15: Adicionando cartucho MySQL.**



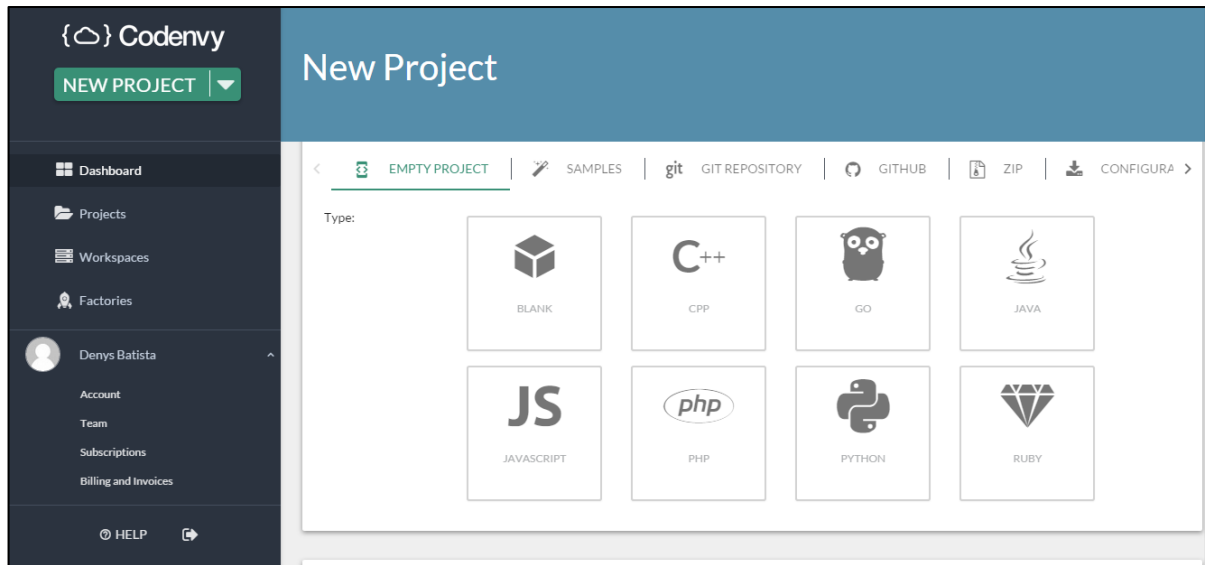
Fonte: Print screen do site no navegador Google Chrome.

Como meio de acesso ao repositório do Openshift, foi criada uma conta gratuita no site da IDE online Codenvy, que é uma IDE na nuvem que permite ao usuário codificar, construir, depurar na nuvem e implantar na PaaS de escolha do usuário. Na IDE foi escolhida a criação de um projeto para a linguagem PHP. Depois da escolha, é carregado o espaço de trabalho da IDE. Podemos perceber esse passo na Figura 16.

Para gerar a conexão SSH do Codenvy com o Openshift, é necessário clicar na barra de menus do espaço de trabalho do projeto na opção *Windows*, depois *Preference* e logo em seguida em *SSH Keystore* e *Generate Key*. No quadro deve ser digitado “rhcloud.com” e logo depois deve ser copiada a chave gerada nesse passo. Com a chave de segurança gerada, foi necessário voltar no console da plataforma Openshift, acessar as configurações e adicionar uma chave publica nas configurações. Nesse espaço deve ser copiada a chave gerada na IDE e clicar em “*Create*” (Figura 17).

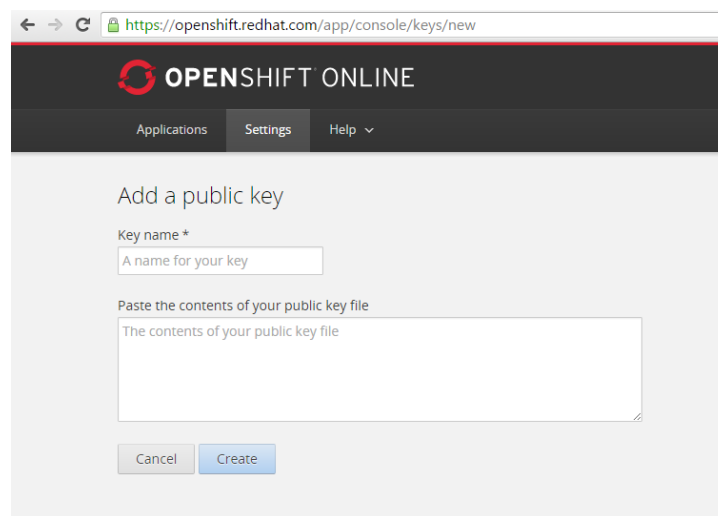
No espaço de trabalho da IDE Codenvy, foi importado o projeto do repositório Openshift seguindo os passos *File > Import > Import From Location > GIT* e foi colado a URL do repositório do projeto no Openshift (Figura 18).

**Figura 16: Criando um novo projeto na IDE Codenvy.**



Fonte: Print screen do site no navegador Google Chrome.

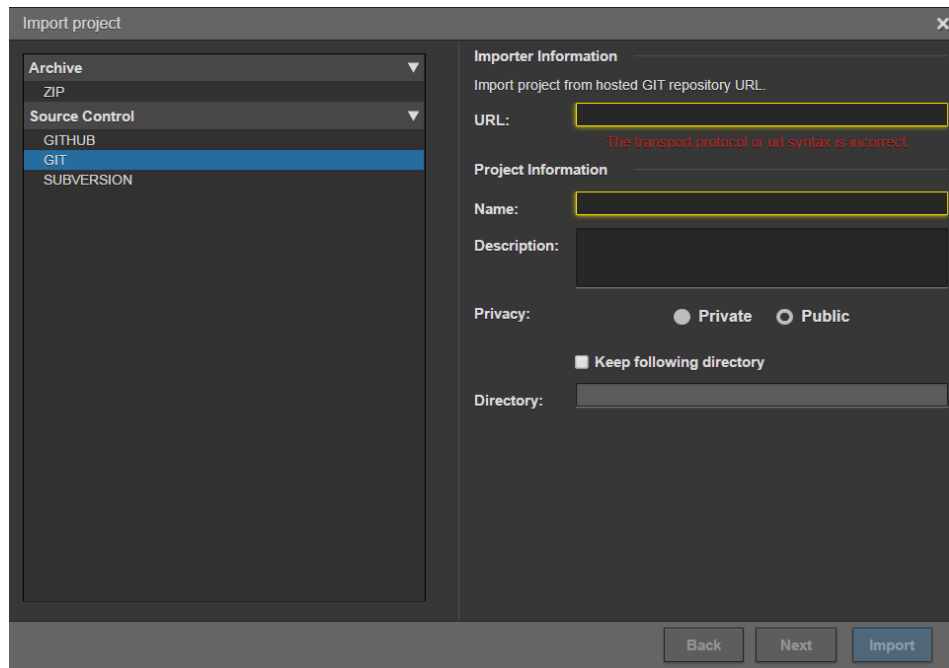
**Figura 17: Adicionado chave pública nas configurações do Openshift**



Fonte: Print screen do site no navegador Google Chrome.

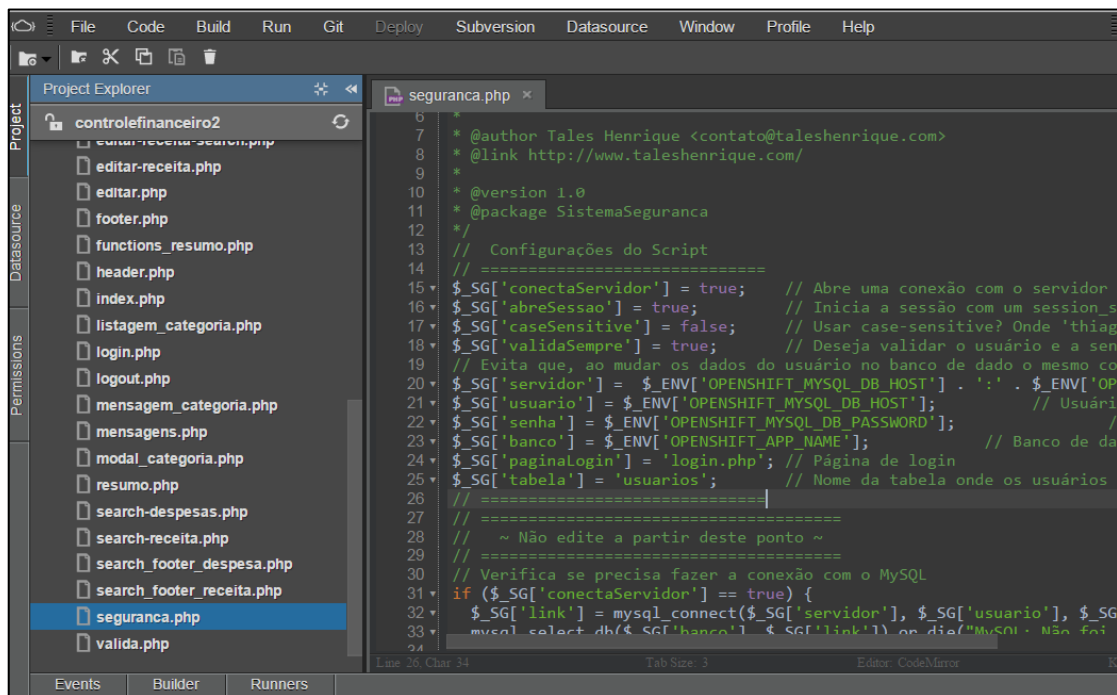
Através da opção “*Upload Folder From Zip*” do menu File da Codenvy, foi possível carregar o código fonte do sistema de controle financeiro localizado na máquina local. Feito isso, foi acessado o arquivo “segurança.php” para alterarmos as informações de conexão com banco de dados, colocando as informações concedidas pela plataforma Openshift ao criar o cartucho MySQL. Foi usado o array `$_ENV` da linguagem PHP para buscar os valores das variáveis de ambiente do Openshift (Figura 19).

Figura 18: Importando projeto do repositório do Openshift.



Fonte: Print screen do site no navegador Google Chrome

Figura 19: Inserindo as variáveis de ambiente no código fonte.



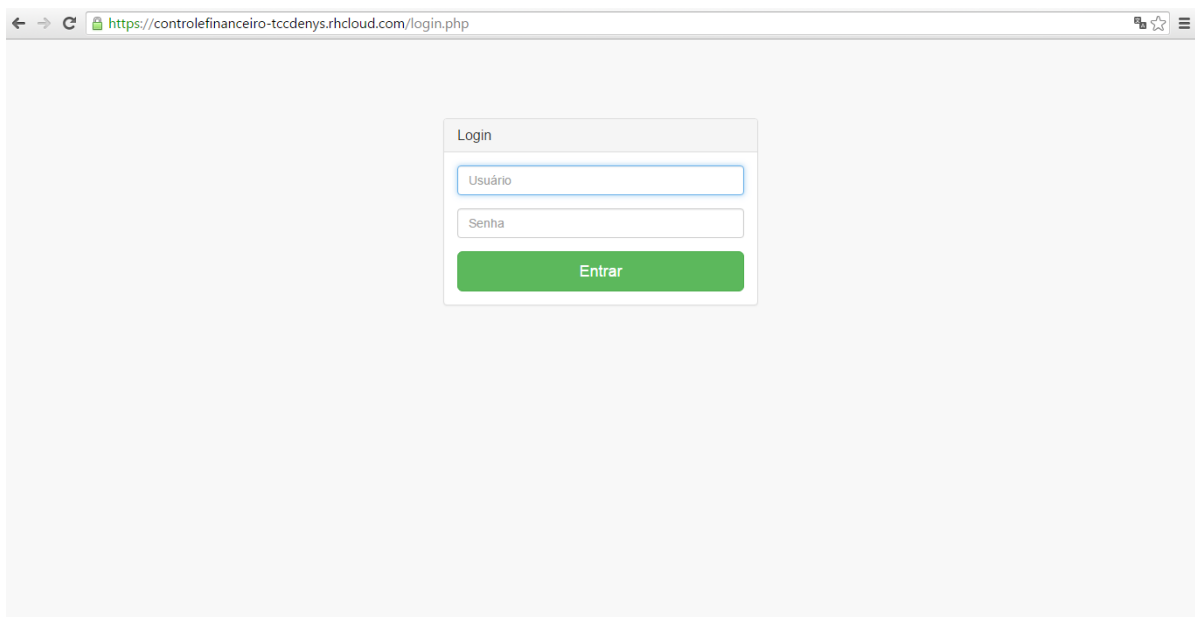
Fonte: Print screen do site no navegador Google Chrome.

Os arquivos do projeto devem ser passados para o repositório do Openshift usando o sistema de controle de versão Git. Para isso, é necessário selecionar os arquivos do projeto, acessar o menu Git, clicar em “Add Index” e confirmar. Em seguida é preciso ir em “Commit”, comentar sobre aquela versão e clicar no botão “Commit”. Por fim, deve ser feito o *push* para

o repositório da plataforma acessando *Git > Remotes > Push*. Com esse processo, a aplicação já está hospedada no Openshift e pode ser acessada através da URL do projeto concedida pelo Openshift. Para importar as tabelas foi utilizado o aplicativo *web phpMyAdmin*.

As Figuras 20 e 21 mostram a aplicação em funcionamento na plataforma OpenShift.

**Figura 20: Página de login da aplicação de controle financeiro no Openshift.**



The screenshot shows a web browser window with the address bar displaying `https://controlefinanceiro-tccdenys.rhcloud.com/login.php`. The main content area features a centered login form titled "Login". The form contains two input fields: "Usuário" and "Senha", followed by a prominent green button labeled "Entrar".

Fonte: Print screen da aplicação no navegador Google Chrome.

**Figura 21: Página de login da aplicação de controle financeiro no Openshift**



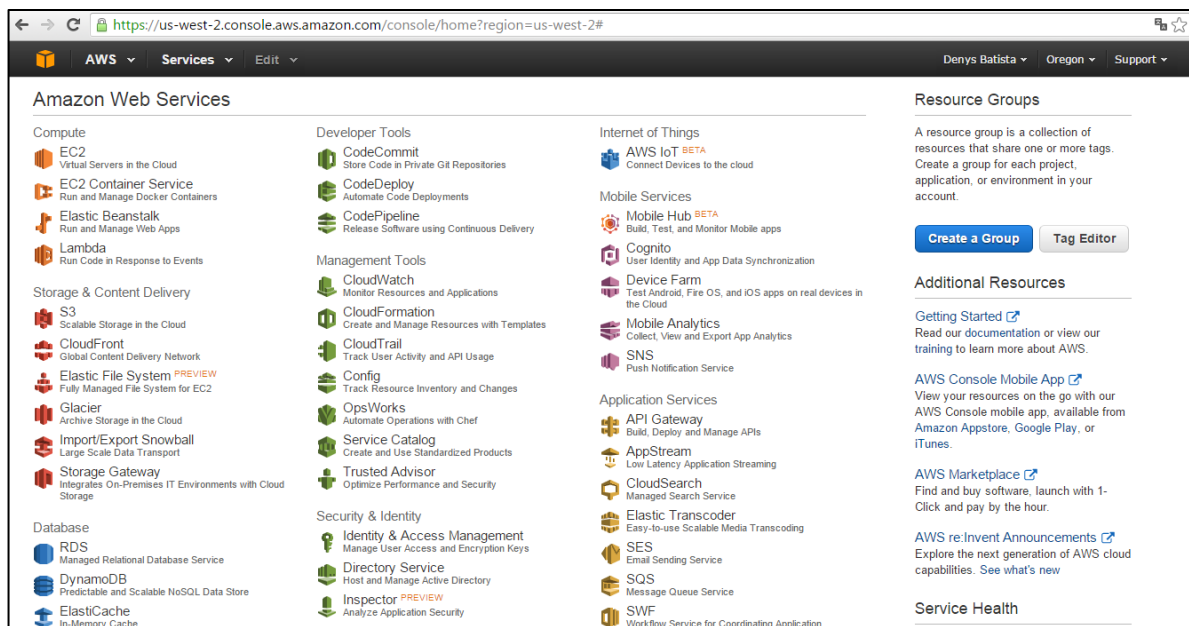
The screenshot displays the main interface of the application, titled "Registro de Movimentos". The top navigation bar includes links for "Página Inicial", "Resumo mensal", "Cadastrar Categoria", and "Buscar", along with the date "Hoje é 28/10/2015" and a user greeting "Olá, Tales". The main content area is divided into two primary sections: "Cadastrar Receita" (highlighted in blue) and "Cadastrar Despesa" (highlighted in red). Each section contains a form with fields for "Valor", "Descrição", "Data", and "Categoria", and a "Cadastrar" button. Below these sections, three summary boxes provide financial data: "Total receita R\$ 300,00", "Total despesa R\$ 100,00", and "Total Mensal R\$ 200,00".

Fonte: Print screen da aplicação no navegador Google Chrome.

### 3.3.2 Implantação do projeto no Elastic Beanstalk da AWS Amazon

No conjunto de serviços em nuvem AWS da Amazon, depois de criada a conta, o usuário pode ter acesso ao console da plataforma (Figura 22). Para criar uma aplicação através do serviço Beanstalk, foi necessário clicar em Elastic Beanstalk e logo depois em *Create New Application*. Em seguida, foi informado o nome da aplicação e feito uma pequena descrição.

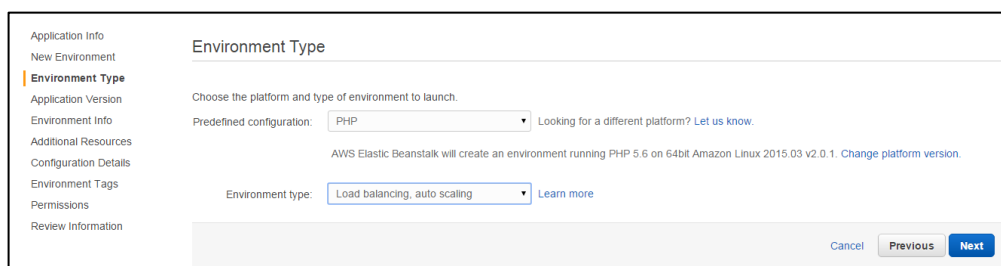
Figura 22: Console da AWS Amazon



Fonte: Print screen da aplicação no navegador Google Chrome.

O Beanstalk oferece ao usuário a oportunidade de escolher o tipo de ambiente de funcionamento da instância. Para esse projeto foi escolhida a opção Web Server e em seguida escolhida a plataforma PHP em um ambiente do tipo auto-escalável (Figura 23). Nesta configuração, a AWS cria uma instância com sistema operacional Amazon Linux 64 bits executando o PHP 5.6.

Figura 23: Escolha do tipo de ambiente do Elastic Beanstalk



Fonte: Print screen da aplicação no navegador Google Chrome.



Em seguida, o usuário pode fazer o *upload* da versão do projeto localizado na máquina local ou escolher iniciar com uma aplicação exemplo do Beanstalk (Figura 24). Foi carregado assim um arquivo com extensão Zip com o projeto do sistema de controle financeiro.

**Figura 24: Fazendo upload da primeira versão da aplicação.**

The screenshot shows the 'Application Version' configuration page in the AWS Management Console. On the left, a navigation menu lists various steps: Application Info, New Environment, Environment Type, Application Version (highlighted), Environment Info, Additional Resources, Configuration Details, Environment Tags, Permissions, and Review Information. The main content area is titled 'Application Version' and contains the instruction 'Select a source for your application version.' Below this, there are three radio button options: 'Sample application' (which is selected), 'Upload your own (Learn more)' (with a blue button labeled 'Escolher arquivo' and the text 'Nenhum arquivo selecionado'), and 'S3 URL' (with a text input field and an example URL 'https://s3.amazonaws.com/s3Bucket/s3Key').

Fonte: Print screen da aplicação no navegador Google Chrome.

Seguindo as etapas de criação de ambiente do *Elastic* Beanstalk, foi criado o nome do ambiente. Este é usado, por padrão, em conjunto com o domínio do serviço “elasticbeanstalk” para criar o endereço de acesso à aplicação. Entretanto, é possível escolher nomes diferentes para o ambiente e para o endereço (Figura 25).

**Figura 25: Etapa de escolha do nome do ambiente**

The screenshot shows the 'Environment Information' configuration page. The left navigation menu is the same as in Figure 24, with 'Environment Info' highlighted. The main content area is titled 'Environment Information' and contains the instruction 'Enter your environment information. Learn more.' Below this, there are three input fields: 'Environment name' (with the value 'controlefinanceiro'), 'Environment URL' (with the value 'controlefinanceiro' and 'elasticbeanstalk.com'), and 'Description' (with a note 'Optional: 200 character maximum'). There is a 'Check availability' button next to the URL field. At the bottom right, there are three buttons: 'Cancel', 'Previous', and 'Next'.

Fonte: Print screen do site no navegador Google Chrome.

Na etapa “*Additional Resources*” foi selecionada a opção para criar uma instância RDS DB junto com o ambiente que está sendo criado, para que seja possível usar um banco de dados. Em seguida, na “*Configuration Details*”, foi selecionado o tipo de instância t2.micro, tipo aconselhado para o uso gratuito, e usado o padrão para as outras configurações (Figura 26). Ainda nos detalhes de configuração o usuário pode selecionar uma chave para acesso remoto. Entretanto não foi selecionada nenhuma chave nessa etapa.

Em “*Environment Tags*” o usuário pode inserir *tags* para ajudar no controle sobre as

informações das instâncias criadas. Para esse trabalho não acrescentado nenhuma *tag*.

**Figura 26: Etapa dos detalhes de configuração**

The screenshot shows the 'Configuration Details' step in the AWS Management Console. The left sidebar contains a navigation menu with options: Application Info, New Environment, Environment Type, Application Version, Environment Info, Additional Resources, **Configuration Details**, Environment Tags, RDS Configuration, VPC Configuration, Permissions, and Review Information. The main content area is titled 'Configuration Details' and includes the following settings:

- Instance type:** A dropdown menu set to 't1.micro'. Below it, a note states: 'Determines the processing power of the servers in your environment.'
- EC2 key pair:** A dropdown menu set to 'Select a key pair' with a 'Refresh' button. Below it, a note states: 'Optional: Enables remote login to your instances.'
- Email address:** An empty text input field. To its right, a note states: 'Optional: Get notified about any major changes to your environment.'
- Application health check URL:** An empty text input field. To its right, a note states: 'Enter the relative URL that ELB continually monitors to ensure your application is available.'
- Enable rolling updates:** A checked checkbox. Below it, a note states: 'Lets you control how changes to the environment's instances are propagated.' A dropdown menu is set to 'Health'. Below that, a note states: 'Specifies whether to wait to deploy updates and deployments according to a set period of time or instance health.'
- Cross zone load balancing:** A checked checkbox. Below it, a note states: 'Enables load balancing across multiple Availability Zones.'
- Connection draining:** A checked checkbox. Below it, a note states: 'Enables the load balancer to maintain connections to an Amazon EC2 instance to complete in-progress requests while also stopping new requests.'
- Connection draining:** A text input field set to '20' followed by 'seconds'.

Fonte: Print screen do site no navegador Google Chrome.

Na etapa “RDS *Configuration*” o usuário pode configurar a instância RDS do banco de dados que será usado pela aplicação (Figura 27). Para o projeto foi escolhida a opção *mysql* na versão 5.6.23, executando em uma instância de classe *db.t2.micro*. Para tamanho de armazenamento, foi escolhido o tamanho mínimo de 5GB. Ainda nessa configuração, deve ser escolhido o nome do usuário e a senha para acesso ao banco de dados.

**Figura 27: Configuração da instância RDS.**

The screenshot shows the 'RDS Configuration' step in the AWS Management Console. The left sidebar contains a navigation menu with options: Application Info, New Environment, Environment Type, Application Version, Environment Info, Additional Resources, Configuration Details, Environment Tags, **RDS Configuration**, Permissions, and Review Information. The main content area is titled 'RDS Configuration' and includes the following settings:

- Snapshot:** A dropdown menu set to 'None' with a 'Refresh' button.
- DB engine:** A dropdown menu set to 'mysql'.
- DB engine version:** A dropdown menu set to '5.6.23'.
- Instance class:** A dropdown menu set to 'db.t1.micro'.
- Allocated storage:** A text input field set to '5' followed by 'GB'. Below it, a note states: 'You must specify a value from 5 GB to 1024 GB.'
- Username:** An empty text input field.
- Password:** An empty text input field.
- Retention setting:** A dropdown menu set to 'Create snapshot'. Below it, a note states: 'Terminating your environment can permanently delete your Amazon RDS DB instance and all its data. By default, AWS Elastic Beanstalk saves a snapshot, which preserves your data but may incur backup storage charges.'
- Availability:** A dropdown menu set to 'Single availability zone'.

Fonte: Print screen do site no navegador Google Chrome.

Na etapa “*Permissions*”, o usuário crie um perfil de instância que usa o recurso IAM para gerenciar as permissões de uso das instâncias (Figura 28). Nessa etapa foi escolhida a opção de uma função já criada.

**Figura 28: Configuração de permissão de uso da instância.**

Fonte: Print screen do site no navegador Google Chrome.

Por fim, “*Review Information*” resume todas as opções que foram escolhidas para a criação do ambiente. O usuário deve clicar em “*Launch*” para iniciar aplicação.

Para configurar a conexão da aplicação com o banco de dados, é necessário voltar ao console da plataforma e escolher a opção RDS. Logo em seguida é preciso clicar em “*instances*” no menu localizado à esquerda da página e selecionar o banco com o nome da aplicação. Nos detalhes da instância é preciso observar se o grupo na opção *Security Groups* é o mesmo criado na configuração de permissões. Observando ainda os detalhes da instância criada, é necessário copiar o endereço *Endpoint* e a porta utilizada, pois são dados essenciais para a conexão com a aplicação (Figura 29).

Usando um editor de texto na máquina local, foram inseridas as informações no arquivo “*segurança.php*” para a conexão com o banco de dados Controlefinanceiro, criado nas configurações do ambiente do Elastic Beanstalk.

Para carregar a nova versão, o usuário deve voltar na Console da plataforma, selecionar a opção “Elastic Beanstalk” e depois clicar no nome da aplicação. No ambiente da aplicação, foi selecionada a opção “*Upload and Deploy*” e buscado o arquivo “*segurança.php*” na máquina local. Em seguida, é preciso dar um nome à versão e clicar em “*Deploy*” (Figura 30). Depois dessa já foi possível acessar a aplicação.

As Figuras 31 e 32 mostram a aplicação em funcionamento no Elastic Beanstalk.

Figura 29: Detalhes da instância RDS.

The screenshot displays the AWS RDS console interface. The left sidebar shows navigation options like 'Instances', 'Reserved Purchases', 'Snapshots', etc. The main content area is titled 'controlefinanceiro' and is divided into several sections:

- Configuration Details:** Engine (MySQL 5.6.23), License Model (General Public License), Created Time (October 23, 2015 at 11:21:09 AM UTC-3), DB Name (controlefinanceiro), Username (awsuser), Option Group (default:mysql-5-6 (in-sync)), Parameter Group (default:mysql5.6 (in-sync)), Copy Tags To Snapshots (No).
- Security and Network:** Availability Zone (us-west-2b), VPC (vpc-1de6b678), Subnet Group (default (Complete)), Subnets (subnet-2761ff7e, subnet-6e410619, subnet-34fcd51), Security Groups (default (sg-210e7a45) (active)), Publicly Accessible (Yes), Endpoint (controlefinanceiro.cm995t356byj.us-west-2.rds.amazonaws.com), Port (3306), Certificate Authority (rds-ca-2015 (Mar 5, 2020)).
- Instance and IOPS:** Instance Class (db.t2.micro), Storage Type (General Purpose (SSD)), IOPS (disabled), Storage (5 GB).
- Encryption Details:** DB Instance Status (available), Multi AZ (No), Automated Backups (Enabled (7 Days)), Latest Restore Time (October 23, 2015 at 11:22:24 AM UTC-3).
- Maintenance Details:** Auto Minor Version Upgrade (Yes), Maintenance Window (tue:13:10-tue:13:40), Backup Window (09:22-09:52), Pending Maintenance (None).

Fonte: Print screen do site no navegador Google Chrome.

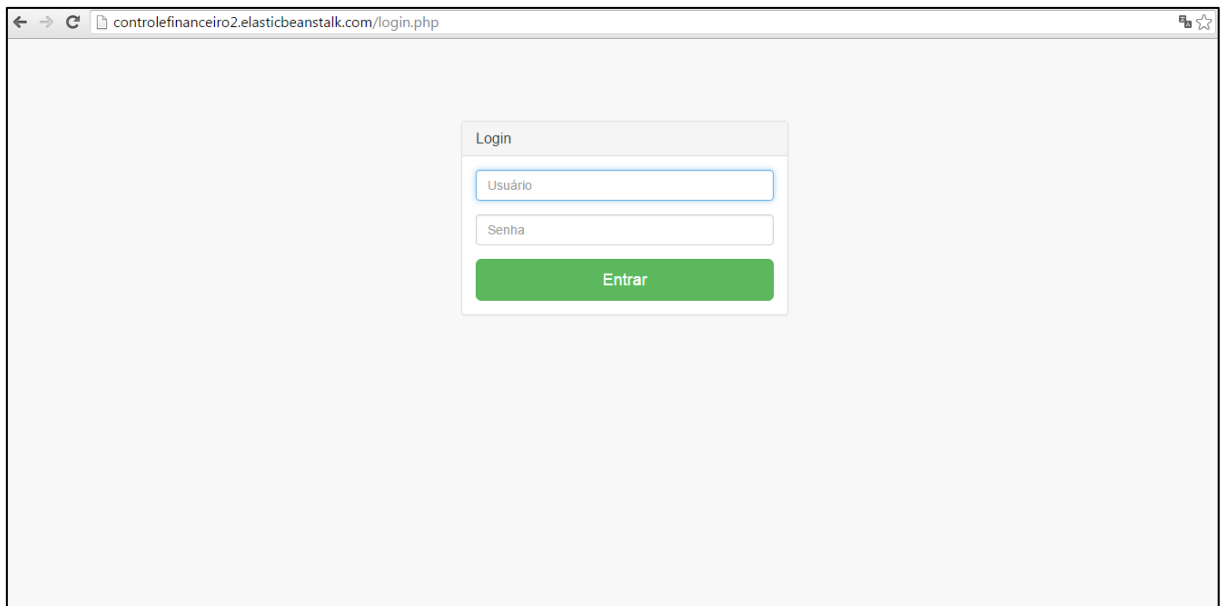
Figura 30: Ambiente da aplicação no Elastic Beanstalk

The screenshot shows the AWS Elastic Beanstalk console. The top navigation bar includes 'AWS', 'Services', 'Edit', and the user 'Denys Batista' in 'Oregon'. The environment name is 'controlefinanceiro2'. The main dashboard displays the following information:

- Health:** A green checkmark icon indicates the environment is 'Ok'. A 'Causes' button is available.
- Running Version:** 'controlefinanceiro'. An 'Upload and Deploy' button is present.
- Configuration:** '64bit Amazon Linux 2015.03 v2.0.1 running PHP 5.6'. A 'Change' button is available.
- Recent Events:** A table showing the following events:
 

Time	Type	Details
2015-10-25 13:24:15 UTC-0300	INFO	Environment health has transitioned from Info to Ok.
2015-10-25 13:23:15 UTC-0300	INFO	Environment health has transitioned from Ok to Info. Command is executing on all instances.
2015-10-25 13:22:50 UTC-0300	INFO	Environment update completed successfully.
2015-10-25 13:22:50 UTC-0300	INFO	New application version was deployed to running EC2 instances.
2015-10-25 13:22:40 UTC-0300	INFO	Deploying new version to instance(s).

Fonte: Print screen do site no navegador Google Chrome.

**Figura 31: Página de login da aplicação implantada na AWS Beanstalk**

The image shows a web browser window with the address bar displaying "controlefinanceiro2.elasticbeanstalk.com/login.php". The main content area features a centered "Login" form. The form has a title "Login" and contains three input fields: "Usuário", "Senha", and a green "Entrar" button.

Fonte: Print screen da aplicação no navegador Google Chrome.

**Figura 32: Página inicial da aplicação implantada na AWS Beanstalk**

The image shows a web browser window with the address bar displaying "controlefinanceiro2.elasticbeanstalk.com/index.php". The page has a navigation menu with items: "Página Inicial", "Resumo mensal", "Cadastrar Categoria", and "Buscar". The date "Hoje é 29/10/2015" and the user name "Oiá, Tales" are displayed in the top right. The main content area is titled "Registro de Movimentos" and contains two forms: "Cadastrar Receita" (blue header) and "Cadastrar Despesa" (red header). Each form has input fields for "Valor", "Descrição", "Data", and "Categoria", and a "Cadastrar" button. Below the forms, there are three summary boxes: "Total receita R\$ 130,00", "Total despesa R\$ 20,00", and "Total Mensal R\$ 110,00".

Fonte: Print screen da aplicação no navegador Google Chrome.

### 3.3.3 Velocidade de acesso

Para medir a velocidade de acesso da página inicial - posterior a página de *login* - da

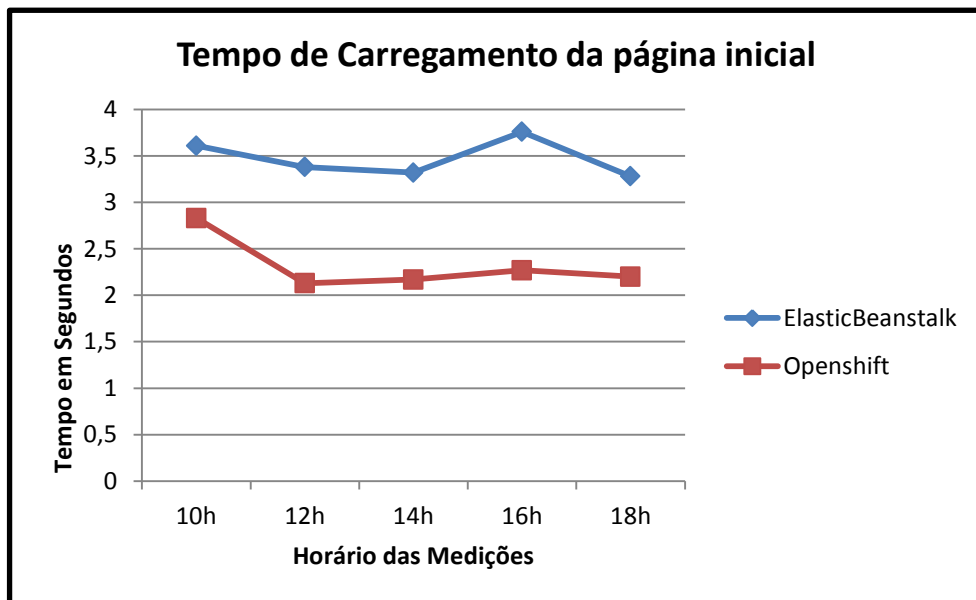
aplicação em cada serviço, foram realizadas cinco requisições no mesmo dia e nos horários 10 h, 12 h, 14 h, 16 h e 18 h. A partir dos dados obtidos foram calculadas as médias e os desvios padrão do tempo de carregamento das páginas. Na Tabela 1 podemos observar os tempos de carregamento de cada serviço.

**Tabela 1: Tempo de carregamento da página inicial em segundos.**

Horários	OpenShift	AWS Elastic Beanstalk
10 H	2,83	3,61
12 H	2,13	3,38
14 H	2,17	3,32
16 H	2,27	3,76
18 H	2,20	3,28
Média	2,32	3,47
Desvio Padrão	0,289655	0,206398

Fonte: Produção do próprio autor.

**Gráfico 1: Tempo de carregamento da página inicial nos dois serviços.**



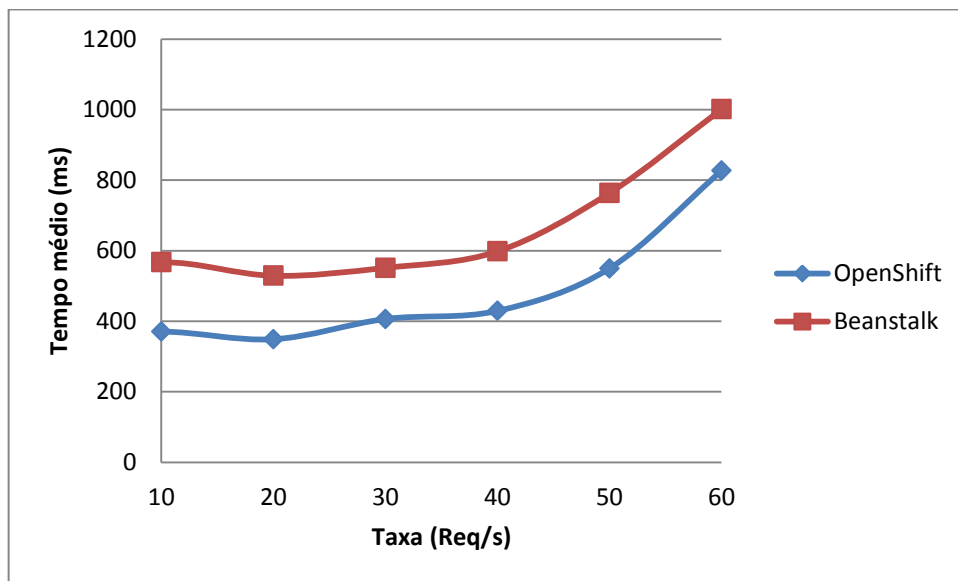
Fonte: Produção do próprio autor.

É possível perceber no Gráfico 1 que em todos os horários em que foram feitas as medições, o acesso à aplicação implantada no OpenShift apresentou uma velocidade razoavelmente maior no carregamento completo da página inicial da aplicação, porém com uma variabilidade um pouco maior que o do Elastic Beanstalk.

### 3.3.4 Desempenho para Várias Requisições por Segundo.

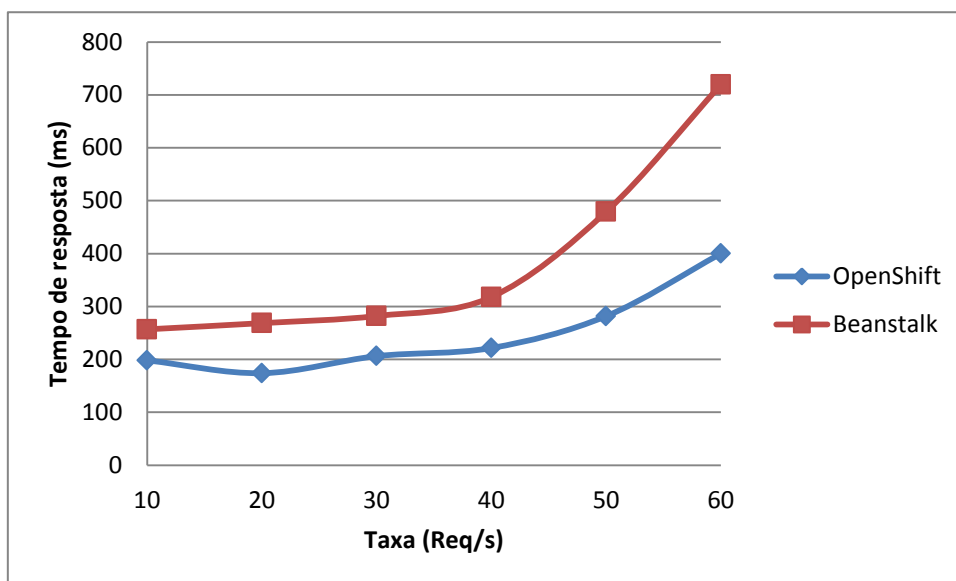
O httpperf é uma ferramenta que permite gerar várias requisições HTTP e medir o desempenho do servidor do ponto de vista dos clientes. Foram feitos seis testes configurados para 100 requisições cada, com as taxas (requisições por segundo) 10, 20, 30, 40, 50 e 60.

**Gráfico 2 – Tempo médio de conexão para diferentes taxas de requisições por segundo.**



Fonte: Produção do próprio autor.

**Gráfico 3: Tempo médio de resposta para diferentes taxas.**



Fonte: Produção do próprio autor.

No Gráfico 2, podemos perceber que o OpenShift oferece tempos médios menores de conexão para várias requisições por segundo e também menores tempos de resposta para diferentes taxas como é possível ver no Gráfico 3,. Isso mostra que a plataforma da Red Hat oferece velocidades maiores de conexão e de resposta do servidor quando há vários clientes acessando a aplicação.

### 3.4 CONCLUSÃO DO ESTUDO COMPARATIVO

Nas pesquisas feitas nesse trabalho é possível perceber que os serviços Openshift Online e o AWS Elastic Beanstalk oferecem facilidades para implantar vários tipos de aplicação. Eles possuem recursos similares, porém se diferenciam em muitos pontos no que diz respeito ao oferecimento gratuito. O 63pshift se destacou na velocidade de acesso das páginas, no desempenho de várias requisições por segundo, na facilidade para acrescentar outros recursos, na grande variedade de linguagens de programação suportadas e pela possibilidade de ser de uso totalmente gratuito. Já o Elastic Beanstalk oferece muitos recursos de segurança, maior número de sistemas operacionais oferecidos e uma gama maior de regiões para criação das instancias. No quadro 4 é mostrado um resumo da comparação feita nesse trabalho.

**Quadro 4: Quadro comparativo dos serviços estudados**

	Openshift	Elastic Beanstalk AWS
<b>Empresa</b>	Red Hat	Amazon
<b>Lançamento</b>	2008	2011
<b>Linguagens de Programação</b>	Ceylon, Go Language, Hack, Java, Node.js, Perl, PHP, Python, Ruby e Scala.	PHP, Java, Python, Ruby, Node.js e .NET.
<b>Banco de dados</b>	MySQL, MongoDB, PostgreSQL, SQLite e Amazon RDS.	MySQL, MongoDB, PostgreSQL, SQLite e Amazon RDS.
<b>Servidores de Aplicação</b>	Jboss AS, Jboss EAP, WildFly, Tomcat, Zend Server e Vert.x.	Apache Tomcat, Apache HTTP, Nginx, Passenger e Microsoft IIS 7.5

Fonte: Produção do próprio autor.



Quadro 4: Continuação

	Openshift	Elastic Beanstalk AWS
<b>Plano gratuito</b>	Doze meses, 750 horas por mês de uso de instância t2.micro	três <i>small gears</i> , cada uma com 512 MB de RAM, 100 MB de swap e 1GB de armazenamento
<b>Sistemas Operacionais Compatíveis</b>	Red Hat Enterprise Linux	Amazon Linux AMI e Windows Server 2008 R2 AMI.
<b>Segurança</b>	Amazon VPC, VPN, IAM	SELinux, Grupos de Controle e Kernel namespaces.
<b>Opções para controle de versão</b>	AWS Management Console, Git e a interface da linha de comando, AWS Toolkit for Visual Studio e o AWS Toolkit for Eclipse	Tecnologia Git
<b>Regiões (Plano Gratuito)</b>	us-east-1	Todas da AWS Amazon
<b>Média do carregamento página inicial</b>	2,32 segundos	3,47 segundos
<b>Desvio padrão carregamento página inicial</b>	0,289655 segundos	0,206398 segundos

Fonte: Produção do próprio autor.

## 4 CONCLUSÃO

Este trabalho apresentou as principais características da computação em nuvem, assim como um pouco da sua história e alguns conceitos e tecnologias relacionadas com este ambiente. Com foco no modelo de serviço (PaaS), foram estudadas algumas importantes plataformas do mercado, mostrando seus principais serviços e recursos e observando a grande importância desse tema para a evolução da informática moderna.

Para o estudo comparativo, foram estudadas a plataforma Openshift da empresa Red Hat e o serviço Elastic Beanstalk da plataforma AWS da Amazon. Foi possível perceber algumas semelhanças entre os serviços, assim como características específicas de cada plataforma. Na implantação da aplicação “Controle Financeiro”, foi possível perceber a facilidade para criação do ambiente e a adição de recursos nos dois serviços. Entretanto, o Openshift apresentou uma velocidade maior no carregamento da página inicial da aplicação e melhor desempenho para várias conexões por segundo. Mesmo com todos os estudos realizados, não é possível definir qual é o melhor, já que depende muito da necessidade do usuário.

Como possíveis trabalhos futuros, se destacam:

- a) O uso de tecnologia Docker em plataformas em nuvem;
- b) estudo comparativo entre Infraestruturas como um Serviço (IaaS);
- c) Comparação entre as principais IDEs em nuvem do mercado.

## 5 REFERÊNCIAS

ALECRIM, Emerson. **O que é cloud computing (computação nas nuvens)?** 2008. Disponível em: <<http://www.infowester.com/cloudcomputing.php>>. Acesso em: 04 jun. 2015.

AMAZON. **Documentação da AWS.** Disponível em: <<https://aws.amazon.com/pt/documentation/>>. Acesso em: 15 out. 2015.

ANDERSON, Chris. **A cauda longa.** Rio de Janeiro: Elsevier, 2006.

AQUINO, Bruno Theodoro de. **Computação em nuvens com Google App Engine e Java.** 2013. 53 f. TCC (Graduação) - Curso de Especialização em Desenvolvimento de Sistemas Para Web, Universidade Estadual de Maringá – UEM, Maringá, 2013.

BUYAYA, R. et al. 2009. **Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility.** Future Gener. Comput. Syst., 25(6):599–616.

CODENVY. **User Guide.** Disponível em: <<http://docs.codenvy.com/>>. Acesso em: 13 out. 2015.

DAVIES, Kathy. **Sobre Máquinas Virtuais do Azure.** 2015. Disponível em: <<https://acom-feature-videos-twitter-card.azurewebsites.net/pt-br/documentation/articles/virtual-machines-about/>>. Acesso em: 30 out. 2015.

DIAS, Lúcio Franco; OLIVEIRA JUNIOR, Edson A.. **Migrando um Sistema de Controle de Protocolos para a Plataforma OpenShift.** 2012. Disponível em: <[http://www.espweb.uem.br/site/files/tcc/2012/Lucio Franco Dias - Migrando um Sistema de Controle de Protocolos para Plataforma OpenShift.pdf](http://www.espweb.uem.br/site/files/tcc/2012/Lucio%20Franco%20Dias%20-%20Migrando%20um%20Sistema%20de%20Controle%20de%20Protocolos%20para%20Plataforma%20OpenShift.pdf)>. Acesso em: 30 out. 2015.

DIOGENES, Yuri; MAUSER, Daniel. **Certificação Security+ - Da Prática Para o Exame SY0-301.** 2ª Edição. Brasil, 2013.

FIREBUG. **Firebug 2.0.13.** Disponível em: <<https://addons.mozilla.org/pt-pt/firefox/addon/firebug/>>. Acesso em: 02 nov. 2015.

GENTZ, Mimi. **Introdução ao Banco de Dados de Documentos do Microsoft Azure.** 2015.

Disponível em: <<https://azure.microsoft.com/pt-br/documentation/articles/documentdb-introduction/>>. Acesso em: 02 nov. 2015.

GEORGE, Andy De. **Should I Choose Cloud Services?** 2015. Disponível em: <<https://azure.microsoft.com/en-us/documentation/articles/cloud-services-choose-me/>>. Acesso em: 30 out. 2015.

GITHUB. **Controle financeiro**. Elaborado por Tales Henrique. Disponível em: <[https://github.com/Tassle/Controle\\_financeiro](https://github.com/Tassle/Controle_financeiro)>. Acesso em: 10 out. 2015.

GOMES, Juliene Cristina. **Estudo de plataformas de computação em nuvens**. 2012. 86 f. TCC (Graduação) - Curso de Sistemas de Informação, Universidade do Planalto Catarinense, Lages, 2012.

GOOGLE. **Google App Engine FAQ**. Disponível em: <<https://cloud.google.com/appengine/kb/?csw=1>>. Acesso em: 20 ago. 2015.

HARVEY, John. **Visão geral do Governo do Microsoft Azure**. 2015. Disponível em: <<https://azure.microsoft.com/pt-br/documentation/articles/azure-government-overview/>>. Acesso em: 30 out. 2015.

HOLLANDA FILHO, Ricardo Luiz Costa. **Uma investigação da plataforma Google App Engine e da migração de aplicações para o Amazon EC2**. 2010. 74 f. Monografia (Especialização) - Curso de Fundação Edson Queiroz Universidade de Fortaleza, Fundação Edson Queiroz Universidade de Fortaleza, Fortaleza, 2010.

KRAMER, Tim. **Security in the Cloud: Is Your PaaS Running on Red Hat Enterprise Linux?**. 2012. Disponível em: <<https://blog.openshift.com/security-in-the-cloud-is-your-paas-running-on-red-hat-enterprise-linux/>>. Acesso em: 20 out. 2015.

LECHETA, Ricardo R.. **AWS para Desenvolvedores: Aprenda a instalar aplicações na nuvem da Amazon AWS**. São Paulo, Sp: Novatec Editora, 2014. 504 p.

LOWE, Janet. **Google**. Rio de Janeiro: Campus, 2009.

MARTINS, Adriano. **Fundamentos de Computação Nuvem para Governos**. Serviço Federal de Processamento de Dados (SERPRO). Brasília, 2010.

MARTINS, Luziane Graciano. **Bibliotecas em Nuvem: o uso da computação em Nuvem em bibliotecas**. Porto Alegre, 2012.

MICROSOFT. **Centro de Documentos**. Disponível em: <<https://azure.microsoft.com/pt-br/documentation/>>. Acessado em 25/09/2015.

NUBLING, Gabriela. **Cloud Computing aplicada ao Cenário Corporativo**. São Paulo, 2011. Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de São Paulo – FATEC-SP, Curso de Tecnologia em Processamento de Dados.

NUNES, Mayza. **História da Computação nas Nuvens**. Disponível em: <[http://www.dsc.ufcg.edu.br/~pet/jornal/agosto2012/matérias/historia\\_da\\_computacao.html](http://www.dsc.ufcg.edu.br/~pet/jornal/agosto2012/matérias/historia_da_computacao.html)>. Acessado em 12/08/2015.

OLIVEIRA, Daniel; VALENTIM JR., Tarcísio; SANTOS, Neide. **Computação em Nuvem para Gerenciamento de Rede de Ensino**. Cadernos do IME: Série Informática: Vol. 31: Junho 2011.

OPENSIFT. **OpenShift Documentation**. Disponível em: <<https://docs.openshift.com/>>. Acesso em: 10 out. 2015.

OPENSIFT b. **Overview of Server Guides**. Disponível em: <<https://developers.openshift.com/en/servers-overview.html>>. Acesso em: 26 out. 2015.

PESSINI, Eduardo Pernambuco; MARTINHO, Julio Cesar Garçon; MAZZOLA, Rafael Martins. **Análise da Plataforma de Desenvolvimento Google App engine**. 2012. 12 f. Curso de Sistemas de Informação, Centro Universitário Padre Anchieta, Jundiaí - SP, 2012. Disponível em: <<http://docslide.com.br/documents/google-app-engine-55c0904e8bcc0.html>>. Acesso em: 29 out. 2015.

PINELI, José Carlos; DUARTE, Mauricio. **Ambiente de desenvolvimento de software em nuvem**. FATEC, Garça, v. 3, n. 10. Disponível em: <[http://www.fatecgarca.edu.br/revista/Volume3/artigos\\_vol3/Artigo\\_10.pdf](http://www.fatecgarca.edu.br/revista/Volume3/artigos_vol3/Artigo_10.pdf)>. Acesso em: 23 jun. 2015.

PISA, Pedro. **O que é e como usar o MySQL?** 2012. Disponível em: <<http://www.techtudo.com.br/artigos/noticia/2012/04/o-que-e-e-como-usar-o-mysql.html>>. Acesso em: 20 out. 2015.

PRADO, Roberto Longhi Rodrigues. A Ampliação do valor de uma empresa através da adoção de um novo modelo: o cloud computing.

RED HAT. **OpenShift Enterprise 2: Deployment Guide**. Disponível em: <[https://access.redhat.com/documentation/en-US/OpenShift\\_Enterprise/2/html-single/Deployment\\_Guide/index.html#Security8](https://access.redhat.com/documentation/en-US/OpenShift_Enterprise/2/html-single/Deployment_Guide/index.html#Security8)>. Acesso em: 30 out. 2015.

RUSCHEL, Henrique; ZANOTTO, Mariana Susan; MOTA, Wélton Costa da. **Computação em Nuvem**. 2010. Especialização em Redes e Segurança de Sistemas, Pontifícia Universidade Católica do Paraná, Curitiba, 2010.

SÃO PAULO (Estado). Secretaria de Gestão Pública. **Computação em Nuvem**. São Paulo. Disponível em: <[http://www.utic.sp.gov.br/paginas/coetic/Computacao\\_Nuvem.html](http://www.utic.sp.gov.br/paginas/coetic/Computacao_Nuvem.html)>. Acessado em 15/08/2015.

SILVA, Diogo. **Google App Engine: Conheça o sistema PaaS da Google**. 2015. Disponível em: <<http://www.cloudmarket.com.br/blog/cloud-computing/google-app-engine-conheca-o-sistema-paas-da-google/>>. Acesso em: 29 out. 2015.

SILVA, Tiago Monteiro Da. **Computação em Nuvem, uma análise das abordagens arquiteturais atuais**. Natal, 2011.

SMITH, Sharon. **Série 8000 StorSimple: uma solução de armazenamento em nuvem híbrida**. 2015. Disponível em: <<https://azure.microsoft.com/pt-br/documentation/articles/storsimple-overview/>>. Acesso em: 15 out. 2015.

SOARES, Rodrigo. **Computação em nuvem com o Google App Engine**. 2009. 65 f. TCC (Graduação) - Curso de Ciência da Computação, Unilasalle, Canoas, 2009.

SOUSA, Flávio R. C.; MOREIRA, Leonardo O.; MACHADO, Javam C. **Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios**. 2009. Disponível em: <<http://www.ufpi.br/subsiteFiles/ercemapi/arquivos/files/minicurso/mc7.pdf>>. Acesso em: 20 ago. 2015.

TAURION, Cezar. **Computação em nuvem**. Rio de Janeiro: Brasport, 2009.

TOSADORE, Rafael de Almeida. **Virtualização: alta disponibilidade, performance e redução de custos**. 2012. 88 f. TCC (Graduação) - Curso de Tecnologia em Redes de Computadores, Faculdade Politec, Santa Bárbara D'oeste, 2012. Disponível em: <<http://grsecurity.com.br/apostilas/Virtualizacao/TCC-Virtualizacao-RafaelTosadore-2012.pdf>>. Acesso em: 13 jun. 2015.

TURKARSLAN, Selcin. **Noções básicas sobre o Banco de Dados SQL do Azure e SQL Server em máquinas virtuais do Azure**. 2015. Disponível em: <<https://azure.microsoft.com/pt-br/documentation/articles/data-management-azure-sql-database-and-sql-server-iaas/>>. Acesso em: 30 out. 2015.

VITTI, Pedro A. F. **Integração do PCMONS com o OpenNebula para Gerência e Monitoramento de Nuvens Privadas**. 2012. 69p. Trabalho de conclusão de curso do curso de Ciência da Computação. UFSC. Florianópolis – SC, 2012.

WESTCON, Equipe. **Entenda o que é o Microsoft Azure e seus principais diferenciais**. 2015. Disponível em: <<http://microsoft.westcon.com/?/post/18/entenda-o-que-e-o-microsoft-azure-e-seus-principais-diferenciais/>>. Acesso em: 30 out. 2015.