

INSTITUTO INFNET
ESCOLA SUPERIOR DE TECNOLOGIA DA INFORMAÇÃO
GRADUAÇÃO EM REDES DE COMPUTADORES



**PROJETO DE BLOCO – ARQUITETURA E
INFRAESTRUTURA DE APLICAÇÕES**
PROJETO FINAL
ALUNO: LUCAS ALONSO SILVA
E-MAIL: lucas.asilva@al.infnet.edu.br
TURMA: GRC - MANHÃ

Sumário

| | |
|---|-----------|
| 1. INTRODUÇÃO..... | 3 |
| 2. PROBLEMA..... | 4 |
| 3. SISTEMAS DE VIRTUALIZAÇÃO | 4 |
| 3.1. VMWARE VSPHERE | 5 |
| 3.2. GOOGLE CLOUD..... | 6 |
| 3.3. DOCKER | 6 |
| 3.4. OPENSTACK | 7 |
| 3.5. AMAZON AWS | 8 |
| 4. APLICAÇÃO: WORDPRESS | 9 |
| 4.1. CARACTERÍSTICAS..... | 9 |
| 4.2. DESENVOLVIMENTO | 10 |
| 4.3. LICENCIAMENTO | 10 |
| 4.4. VERSIONAMENTO | 10 |
| 4.6. RECURSOS..... | 10 |
| 4.7. REQUISITOS | 11 |
| 5. PROJETO | 12 |
| 5.1. INFRAESTRUTURA | 13 |
| 5.1.1. Wordpress..... | 13 |
| 5.1.2. Amazon AWS..... | 13 |
| 5.2. ESTIMATIVA DE RECURSOS | 14 |
| 5.3. ESTIMATIVA DE CUSTOS | 15 |
| 5.4. CRONOGRAMA ESTIMADO | 15 |
| 5.5. PLANO DE IMPLEMENTAÇÃO | 15 |
| 5.5.1. Infraestrutura Amazon AWS..... | 16 |
| 5.5.2. Aplicação..... | 23 |
| 6. CONCLUSÕES | 39 |
| 7. REFERÊNCIAS | 41 |

Resumo. Este documento tem o objetivo de apresentar a escolha de uma aplicação a ser implementada em uma infraestrutura de nuvem, assim como a plataforma de virtualização utilizada para isso, com base nos conhecimentos adquiridos na disciplina de Arquitetura e Infraestrutura de Aplicações. Ao longo deste projeto, iremos expor de forma detalhada as características de desenvolvimento, infraestrutura, implantação e configuração da aplicação escolhida e apresentar um projeto desta implementação em um sistema totalmente virtualizado.

1. Introdução

Atualmente, os sistemas de informação tem papel indispensável para o bom funcionamento de diversos processos em todos os tipos de empresa. Sendo assim, cada vez mais empresas estão buscando formas de reduzir os custos e complexidade com o ambiente de TI, automatizando processos e otimizando os recursos disponíveis.

A virtualização ou “*cloud computing*” é uma ótima solução para atingir estes objetivos. “*Virtualização é o processo de criar uma representação baseada em software (ou virtual) de algo, em vez de um processo físico. A virtualização pode se aplicar a aplicativos, servidores, armazenamento e redes, é a maneira mais eficaz de reduzir as despesas de TI e, ao mesmo tempo, aumentar a eficiência e a agilidade para empresas de todos os portes.*”¹

Desta forma, ao longo deste documento, iremos apresentar prática e teoricamente os conceitos de virtualização implementando uma aplicação distribuída em servidores virtualizados utilizando uma plataforma de nuvem que será escolhida ao longo do projeto.

¹ <https://www.vmware.com/br/solutions/virtualization.html>

2. Problema

A empresa LAS Technologies é uma empresa do ramo da Tecnologia da Informação, desenvolvendo soluções, com diversas finalidades, específicas para seus clientes. Ainda sendo uma empresa de pequeno porte, está em crescimento. Atualmente a empresa conta com cinco principais áreas: desenvolvimento, infraestrutura, recursos humanos, financeiro e marketing.

Os diretores da empresa tem vontade de atualizar e migrar maior parte de seu datacenter para nuvem pois é uma ótima tecnologia para reduzir custos e volume de trabalhos, além de ser relativamente nova e, com isso, poderíamos dominá-la e incluí-la em nosso catálogo de serviços. A virtualização provê alta integridade, disponibilidade e confidencialidade para a rede, principais pilares da segurança da informação. A facilidade e rapidez de implementação de máquinas virtuais torna o datacenter altamente escalável, garantindo o bom funcionamento de seus processos, o que seria um mais um ponto positivo ao migrar dados para a nuvem.

Nosso objetivo inicialmente é projetar e implementar uma aplicação hospedada, distribuídamente, em uma plataforma de nuvem. Escolhida a partir de uma análise com as principais ferramentas de virtualização do mercado, atualmente. Com o sucesso deste projeto poderemos propor uma migração do datacenter atual e implementação de novas aplicações, como sistemas de monitoramento, service desk, etc.

3. Sistemas de Virtualização

Atualmente, existem diversos sistemas de virtualização e infraestrutura de nuvem disponíveis no mercado. Como os principais podemos citar o VMWare vSphere, Google Cloud Computing, Microsoft Azure, Amazon AWS. Existem também, projetos open-source como o Docker e OpenStack. Apesar de serem diferentes projetos e terem suas singularidades, por exemplo, suas interfaces de administração, a usabilidade e forma de interação com o usuário. Todos possibilitam a criação de máquinas virtuais, ou seja, um software, para simular

um equipamento físico, hardware. Além disso, alguns projetos tem funcionalidades muito parecidas. Por exemplo: na Amazon AWS nós temos o serviço de computação, chamado Elastic Compute Cloud (EC2), usado para criar instancias, especificando a memória utilizada, tamanho do storage, etc, enquanto no Openstack possuímos o Nova, que possui a mesma finalidade. O Openstack é um sistema operacional de nuvem composto por diversos serviços. A VMWare possui o pacote vSphere que também possui um sistema operacional de nuvem, o ESXi. Estes dois projetos ainda possuem outro serviço a ser comparado. O VMWare vCenter pode ser comparado ao Horizon do Openstack, pois ambos são interfaces de gerenciamento do ambiente de nuvem. A seguir veremos uma breve descrição destas principais ferramentas.

3.1. VMWare vSphere

“VMware vSphere® uses virtualization to transform individual data centers into aggregated computing infrastructures that include CPU, storage, and networking resources. VMware vSphere manages these infrastructures as a unified operating environment and provides you with the tools to administer the data centers that participate in that environment.

The VMware vSphere stack comprises virtualization, management, and interface layers. The two core components of vSphere are ESXi and vCenter Server. ESXi is the virtualization platform where you create and run virtual machines and virtual appliances. vCenter Server is the service through which you manage multiple hosts connected in a network and pool host resource”.²

O VMWare vSphere é uma pilha de software composta por dois principais componentes: o ESXi e o vCenter. O primeiro é de fato o sistema operacional de nuvem da VMWare. O segundo é o serviço de gerenciamento de todo o ambiente criado.

² <https://docs.vmware.com/en/VMware-vSphere/index.html>

3.2. Google Cloud

“Google Compute Engine delivers virtual machines running in Google's innovative data centers and worldwide fiber network. Compute Engine's tooling and workflow support enable scaling from single instances to global, load-balanced cloud computing.

Compute Engine's VMs boot quickly, come with persistent disk storage, and deliver consistent performance. Our virtual servers are available in many configurations including predefined sizes or the option to create Custom Machine Types optimized for your specific needs”.³

O sistema de virtualização do google segue o mesmo padrão de outras ferramentas como o openstack e a Amazon AWS, possuem um serviço de computação que possibilita a criação rápida e simples de instancia, que nada mais são do que máquinas virtuais. Entre outros inúmeros serviços que podem ser vistos no site do projeto: <https://cloud.google.com/?hl=pt-br>.

3.3. Docker

“Docker is the world's leading software container platform. Developers use Docker to eliminate “works on my machine” problems when collaborating on code with co-workers. Operators use Docker to run and manage apps side-by-side in isolated containers to get better compute density. Enterprises use Docker to build agile software delivery pipelines to ship new features faster, more securely and with confidence for both Linux and Windows Server apps”.⁴

O docker é um projeto open-source e seus colaboradores tem trabalhado para acabar com o problema de compatibilidade de algumas aplicações. Ele utiliza um sistema de containers. Estes são um pedaço de software, diferente de máquinas virtuais que agrupam um sistema operacional inteiro, que disponibilizam um ambiente para rodar uma aplicação de forma horizontal no

³ <https://cloud.google.com/compute/>

⁴ <https://www.docker.com/what-docker>

sistema. Disponibiliza as bibliotecas necessárias para rodar a aplicação e nada mais. Possibilitando a execução de apenas uma aplicação virtualizada e não um sistema operacional inteiro. O que também não é impossível de fazer, o docker também suporta a virtualização de um sistema operacional, ou seja, possibilita a criação de máquinas virtuais. Muitas vezes com apenas uma linha de comando em um terminal. Muito prático.

3.4. Openstack

“OpenStack software controls large pools of compute, storage, and networking resources throughout a datacenter, managed through a dashboard or via the OpenStack API. OpenStack works with popular enterprise and open source technologies making it ideal for heterogeneous infrastructure.

Hundreds of the world’s largest brands rely on OpenStack to run their businesses every day, reducing costs and helping them move faster. OpenStack has a strong ecosystem, and users seeking commercial support can choose from different OpenStack-powered products and services in the Marketplace.

The software is built by a thriving community of developers, in collaboration with users, and is designed in the open at our Summits”.⁵

Openstack, assim como o Docker, é um projeto open-source, mantido por seus colaboradores e sua comunidade. É uma ferramenta muito utilizada no mundo inteiro, por grandes marcas, para hospedar seus serviços na nuvem. Ele possibilita a criação de um ambiente de nuvem privada altamente funcional, com a possibilidade de configurações avançadas de redes, autenticação, armazenamento, gerenciamento, com seus inúmeros serviços desenvolvidos por seus colaboradores. Dentre eles: Nova, Swift, Keystone, Neutron, Horizon, Glance, etc.

⁵ <https://www.openstack.org/>

3.5. Amazon AWS

*“A Amazon Web Services (AWS) é uma plataforma de serviços em nuvem segura, oferecendo poder computacional, armazenamento de banco de dados, distribuição de conteúdo e outras funcionalidades para ajudar as empresas em seu dimensionamento e crescimento”.*⁶

A Nuvem AWS possibilita a criação de soluções para construir aplicações sofisticadas com mais flexibilidade, escalabilidade e confiabilidade.

*“A execução das suas soluções na Nuvem AWS pode ajudar a disponibilizar suas aplicações em produção com maior rapidez, sem deixar de oferecer o mesmo nível de segurança desfrutado por organizações como Pfizer, Intuit e Marinha dos EUA. A AWS também oferece recursos em todo o mundo, o que permite que você implante as soluções onde os clientes estão. A Nuvem AWS disponibiliza facilmente um amplo conjunto de serviços, parceiros e opções de suporte para ajudar a assegurar que você se concentre nos fatores que garantirão o sucesso da sua solução”.*⁷

A AWS possui inúmeros serviços para serem executados em sua nuvem, como o EC2, VPC, Route53, RDS, ELB, EBS, etc. Muitos podem ser comparados à outras plataformas de virtualização como o openstack. Por exemplo, o serviço EC2 se equivale ao nó Nova do openstack. Ambos de computação, permitem a criação de instancias a partir de imagens de sistemas operacionais para criar máquinas virtuais. Outra equivalência poderia ser o VPN da Amazon com o Neutron, do Openstack. Ambos gerenciadores de infraestrutura de redes. Permitem a criação de redes separadas logicamente, como redes internas, vlans, além da criação de gateways para rotear o tráfego quando necessário e permitido. O sistema de armazenamento de bloco, RDS da AWS com Swift do Openstack, entre outros.

Apesar da Amazon ser um serviço pago, de licença fechada, é uma ótima alternativa como plataforma de virtualização. Seu sistema de pagamento é a

⁶ <https://aws.amazon.com/pt/what-is-aws/>

⁷ <https://aws.amazon.com/pt/solutions/>

partir dos serviços que forem utilizados. Caso não tenha serviços sendo utilizados, o valor pago seria US\$0,00. Inclusive a Amazon disponibiliza uma calculadora online (<https://calculator.s3.amazonaws.com/index.html>) para orçar seus projetos, antes de serem de fato executados.

Com a garantia de suporte e qualidade do serviço graças ao seu número imenso de parcerias, esta opção se torna muito atraente para hospedar um ambiente de nuvem íntegro, escalável e altamente disponível.

4. Aplicação: Wordpress

A aplicação escolhida para desenvolvermos este projeto de website hospedado em nuvem foi o Wordpress. Sendo uma ferramenta Open-Source (distribuída sob a licença GPLv2), foi uma solução barata e de rápida implementação. *“WordPress é um aplicativo de sistema de gerenciamento de conteúdo para web, escrito em PHP com banco de dados MySQL, voltado principalmente para a criação de sites e blogs via web.”*⁸ Além disso, o wordpress permite a instalação de diversos plugins que o tornam uma aplicação muito mais completa e funcional.

4.1. Características

Algumas das características desta aplicação são:

- Desenvolvido em PHP;
- Backend com banco de dados MySQL;
- Front-end com servidor WEB Apache;

⁸ <https://pt.wikipedia.org/wiki/WordPress>

4.2. Desenvolvimento

O desenvolvimento desta aplicação é Open-Source feito pela sua comunidade na internet, apesar de estar associada a empresa *Automattic*, onde diversos desenvolvedores desta ferramenta são funcionários. O Wordpress possui um browser onde o usuário pode visualizar, baixar e enviar requisições de desenvolvimento para correções de bugs, criação de plugins, etc. no endereço:

<https://code.trac.wordpress.org/browser>

4.3. Licenciamento

O software Wordpress é um software de código aberto, ou seja, Open Source, distribuído sob a licença GPLv2. Sem nenhum custo de licenciamento.

4.4. Versionamento

“The built WordPress source, licensed under the GNU General Public License version 2 (or later), can be browsed online or checked out locally with Subversion or Git”.⁹

O código fonte do Wordpress pode ser visto online ou pode ser baixado pelo git, que é o principal sistema de versionamento da aplicação:

Git mirror: [git://core.git.wordpress.org/](https://core.git.wordpress.org/)

4.6. Recursos

Alguns dos recursos incluem:

- *Gerar XML, XHTML, e CSS em conformidade com os padrões W3C*
- *Gerenciamento integrado de ligações*
- *Estrutura de permalink amigável aos mecanismos de busca*

- *Suporte extensivo a plug-ins*
- *Categorias aninhadas e múltiplas categorias para artigos*
- *TrackBack e Pingback*
- *Filtros tipográficos para formatação e estilização de texto corretas*
- *Páginas estáticas*
- *Múltiplos autores*
- *Suporte a tags (desde a versão 2.3)*
- *Pode gerenciar múltiplos blogs em subpastas ou subdomínios (desde a versão 3.0)*
- *Importação e exportação de dados*
- *API de desenvolvimento de plugins*
- *Níveis, promoção e rebaixamento de usuários*
- *Campos personalizados que permitem armazenar dados extras no banco de dados*¹⁰

4.7. Requisitos

De acordo com o site da comunidade wordpress os principais requisitos do sistema são:

Recursos básicos:

- Servidor baseado em UNIX/Linux1
- PHP versão 5.2.4 ou superior
- MySQL versão 5.0 ou superior

⁹ <https://wordpress.org/download/source/>

¹⁰ <https://pt.wikipedia.org/wiki/WordPress>

- Memória para o PHP de pelo menos 64 MB (Somente para o software WordPress, sem plugins adicionais)

Recursos extras:

- Memória para o PHP de pelo menos 256 MB²
- Apache ou Nginx
- Módulo mod_rewrite do Apache ativo
- Extensões PHP como php_exif, php_GD etc (recursos nativos e de plugins)¹¹

Como podemos ver, o wordpress possui uma série de recursos que o tornam uma ferramenta de boa qualidade para hospedar o site da empresa e ser o primeiro projeto baseado em nuvem da empresa.

5. Projeto

Neste projeto, optamos por usar a plataforma de nuvem da Amazon AWS para implementar nossa nuvem privada e hospedar o website. Levamos em conta questões de praticidade, confiabilidade, escalabilidade e segurança.

O projeto será dividido em quatro partes. Primeiro planejaremos a infraestrutura da aplicação dentro do ambiente escolhido, em seguida faremos uma estimativa dos recursos necessários, assim como do custo. Por fim, apresentaremos o plano de implementação da solução.

¹¹ https://codex.wordpress.org/pt-br:Requisitos_do_WordPress

5.1. Infraestrutura

5.1.1. Wordpress

A infraestrutura da aplicação Wordpress é baseada em cliente servidor. Onde o cliente faz requisições via web browser para o servidor frontend, que por sua vez repassa a requisição para o servidor de banco de dados MySQL. Este devolve a informação para o servidor apache que a entrega ao cliente novamente, fechando a comunicação. O diagrama a seguir representa as etapas dessa comunicação.

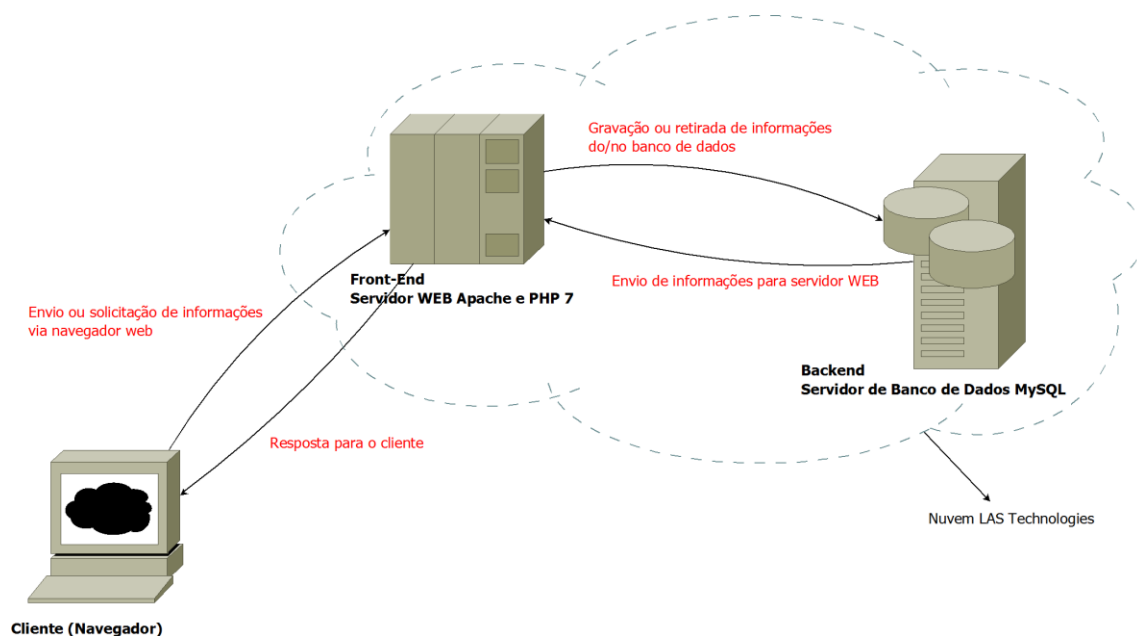


Figura 1. Diagrama de Infraestrutura do Wordpress

5.1.2. Amazon AWS

Agora que entendemos como é a infraestrutura do wordpress, precisamos projetá-la dentro da estrutura de nuvem da amazon AWS. Inicialmente, vamos precisar de dois servidores Linux e um ambiente de nuvem privada.

Sendo assim, na linguagem da Amazon, necessitamos de duas instancias EC2 (Elastic Cloud Computing), com servidores Ubuntu Linux 16.04 LTS, com EBS (Elastic Block Storage) de 500GB cada e uma VPC (Virtual Private Cloud) com

uma rede /16, uma sub-rede /24 e um gateway de acesso a internet com as rotas necessárias. De acordo com o diagrama proposto a seguir.

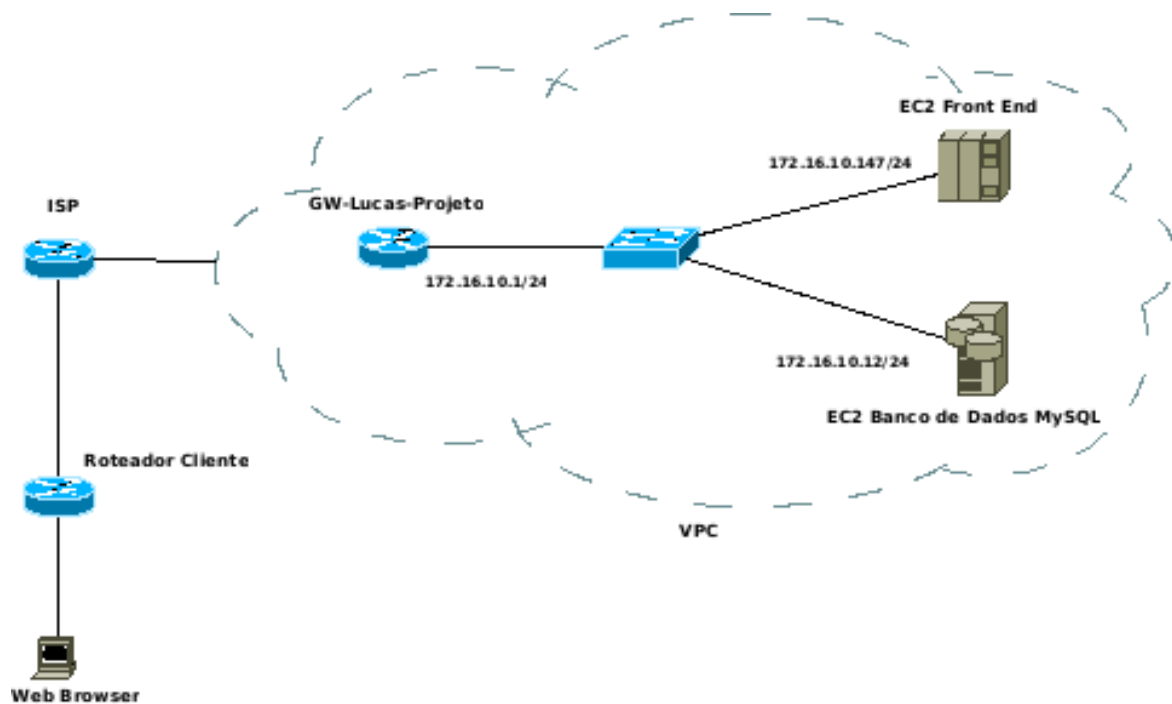


Figura 2. Diagrama de Infraestrutura da AWS

5.2. Estimativa de Recursos

Como dito anteriormente, iremos utilizar um sistema de nuvem privada com dois servidores virtualizados: um web server e um banco de dados. Dessa forma precisaremos de um sistema de virtualização e dois servidores linux ubuntu server virtualizados:

- **Sistema de virtualização:** Amazon AWS;
- **Servidor WEB virtualizado:** Servidor Linux Ubuntu Server com Apache WEB Server e PHP na última versão estável;
- **Servidor de Banco de Dados:** Servidor Linux Ubuntu Server com banco de dados MySQL na última versão estável

- **Hardware:** Equivalente para que cada servidor virtualizado possua 4096MB de memória RAM, 2 CPU, espaço de armazenamento de 500GB e uma interface de rede Gigabit Ethernet.

5.3. Estimativa de Custos

Os custos aproximados do projeto podem ser calculados no site da amazon como base. Utilizando os requisitos estipulados acima: duas instancias EC2 e uma VPC deu um valor de US\$202,42. O cálculo pode ser visto em: <http://calculator.s3.amazonaws.com/index.html#r=IAD&key=calc-6AFD5D5B-218A-43E5-9BD7-1238B0F07EDA>

5.4. Cronograma Estimado

A implementação desta solução é simples e rápida. Mas para garantir uma alta qualidade neste processo seria melhor estimarmos o tempo de implementação em **uma semana**, podendo aumentar, a partir do cadastro no amazon AWS, sendo:

- 1 dia para estudo e implementação do VPC
- 1 dia para estudo e implementação das instancias EC2
- 2 dias para desenvolvimento do código Ansible e implementação da aplicação
- 3 dias para configuração do site (tempo variável de acordo com o departamento de desenvolvimento)

5.5. Plano de Implementação

O processo de implementação de uma nova tecnologia pode ser algo bem complexo. Com isso planejaremos cada etapa deste processo. Iniciaremos implementando a infraestrutura de nuvem na amazon AWS, com todos os

serviços necessários. Após iniciar as instancias EC2 faremos a instalação da aplicação Wordpress utilizando o software de gerenciamento de configuração Ansible. Com este código YAML, linguagem utilizada pelos scripts ansible, poderemos implementar esta aplicação de maneira automatizada, muito rápida e sem nenhum esforço em outras instancias, caso seja necessário futuramente.

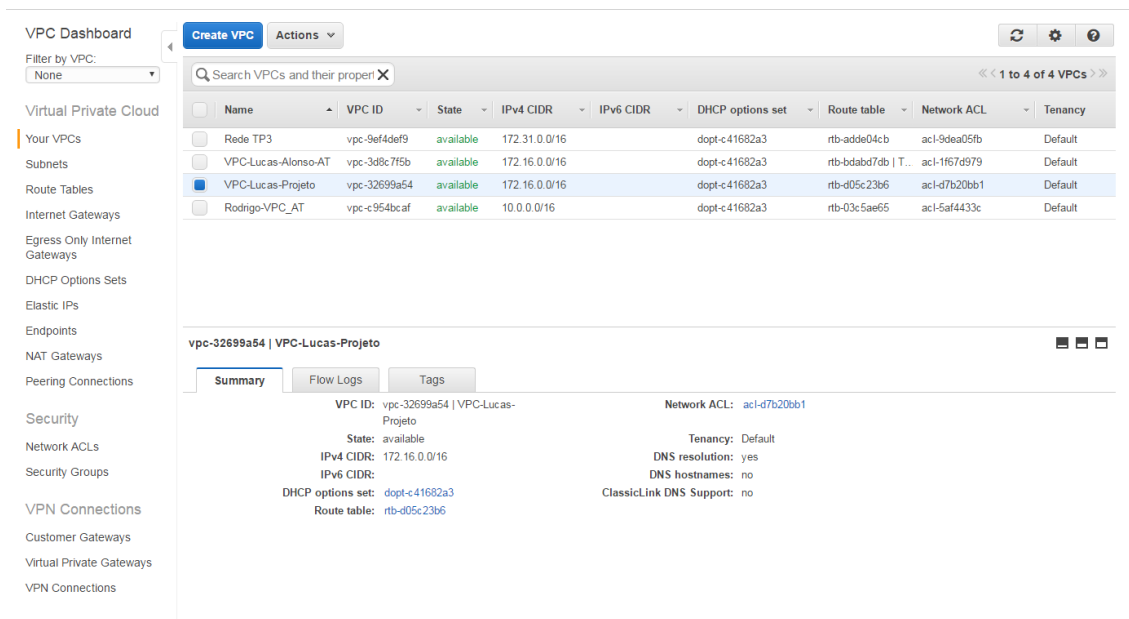
5.5.1. Infraestrutura Amazon AWS

Como vimos na Figura 2 (Diagrama de Infraestrutura na AWS), será necessário implementar dois principais serviços da Amazon: o EC2 e o VPC.

5.5.1.1. Virtual Private Cloud

Começaremos pela configuração da infraestrutura de nuvem privada.

Primeiramente criamos a VPC com o endereçamento 172.16.0.0/16.



VPC Dashboard

Create VPC Actions

Filter by VPC: None

Search VPCs and their properties

| Name | VPC ID | State | IPv4 CIDR | IPv6 CIDR | DHCP options set | Route table | Network ACL | Tenancy |
|---------------------|--------------|-----------|---------------|-----------|------------------|---------------------|--------------|---------|
| Rede TP3 | vpc-9ef4def9 | available | 172.31.0.0/16 | | dopt-c41682a3 | rtb-adde04cb | acl-9dea05fb | Default |
| VPC-Lucas-Alonso-AT | vpc-3d8c7f5b | available | 172.16.0.0/16 | | dopt-c41682a3 | rtb-bdabd7db T... | acl-1f67d979 | Default |
| VPC-Lucas-Projeto | vpc-32699a54 | available | 172.16.0.0/16 | | dopt-c41682a3 | rtb-d05c23b6 | acl-d7b20bb1 | Default |
| Rodrigo-VPC_AT | vpc-c954bcac | available | 10.0.0.0/16 | | dopt-c41682a3 | rtb-03c5ae65 | acl-5af4433c | Default |

vpc-32699a54 | VPC-Lucas-Projeto

Summary Flow Logs Tags

VPC ID: vpc-32699a54 | VPC-Lucas-Projeto

State: available

IPv4 CIDR: 172.16.0.0/16

IPv6 CIDR:

DHCP options set: dopt-c41682a3

Route table: rtb-d05c23b6

Network ACL: acl-d7b20bb1

Tenancy: Default

DNS resolution: yes

DNS hostnames: no

ClassicLink DNS Support: no

Figura 3. Criação da VPC-Lucas-Projeto

Criada a rede, criamos uma sub-rede menor, 172.16.10.0/24, onde estarão localizadas nossas duas instancias EC2.

VPC Dashboard

Filter by VPC: None

Virtual Private Cloud

Your VPCs

Subnets

Route Tables

Internet Gateways

Egress Only Internet Gateways

DHCP Options Sets

Elastic IPs

Endpoints

NAT Gateways

Peering Connections

Security

Network ACLs

Security Groups

VPN Connections

Customer Gateways

Virtual Private Gateways

VPN Connections

Create Subnet Subnet Actions

Search Subnets and their prop X

<< 1 to 8 of 8 Subnets >>

| Name | Subnet ID | State | VPC | IPv4 CIDR | Available IPv4 | IPv6 CIDR | Availability Zone |
|----------------------|-----------------|-----------|------------------------------------|----------------|----------------|-----------|-------------------|
| Public subnet A | subnet-14515573 | available | vpc-c954bc4f Rodrigo-VPC_AT | 10.0.0.0/24 | 249 | | us-west-2a |
| Lucas-Subnet-2C | subnet-a9d97bf2 | available | vpc-3d8c7f5b VPC-Lucas-Alonso-AT | 172.16.10.0/24 | 250 | | us-west-2c |
| Public subnet B | subnet-ba0e51f3 | available | vpc-c954bc4f Rodrigo-VPC_AT | 10.0.1.0/24 | 248 | | us-west-2b |
| Lucas-Subnet-2A | subnet-366d6151 | available | vpc-3d8c7f5b VPC-Lucas-Alonso-AT | 172.16.20.0/24 | 249 | | us-west-2a |
| | subnet-e4831bad | available | vpc-9ef4def9 Rede TP3 | 172.31.32.0/20 | 4089 | | us-west-2b |
| Subnet-Lucas-Projeto | subnet-dcf3ffbb | available | vpc-32699a54 VPC-Lucas-Projeto | 172.16.10.0/24 | 249 | | us-west-2a |
| | subnet-6cfeb80b | available | vpc-9ef4def9 Rede TP3 | 172.31.16.0/20 | 4091 | | us-west-2a |
| | subnet-b7d326ec | available | vpc-9ef4def9 Rede TP3 | 172.31.0.0/20 | 4090 | | us-west-2c |

subnet-dcf3ffbb | Subnet-Lucas-Projeto

Summary Route Table Network ACL Flow Logs Tags

Subnet ID: subnet-dcf3ffbb | Subnet-Lucas-Projeto Availability Zone: us-west-2a

IPv4 CIDR: 172.16.10.0/24 Route table: rtb-a6f11ec8 | Routes-Lucas-Projeto

IPv6 CIDR: State: available Network ACL: acl-d7b20bb1

VPC: vpc-32699a54 | VPC-Lucas-Projeto Default subnet: no

Available IPs: 249 Auto-assign Public IP: no

Auto-assign IPv6 address: no

Figura 4. Criação da sub-rede Subnet-Lucas-Projeto

Após a criação da sub-rede criamos um gateway de acesso a internet para nossas redes. Pois como vamos hospedar um website, o mesmo deve poder ser acessado e enviar informações através do protocolo HTTP, na internet.

VPC Dashboard

Filter by VPC: None

Virtual Private Cloud

- Your VPCs
- Subnets
- Route Tables
- Internet Gateways**
- Egress Only Internet Gateways
- DHCP Options Sets
- Elastic IPs
- Endpoints
- NAT Gateways
- Peering Connections
- Security
- Network ACLs
- Security Groups
- VPN Connections
- Customer Gateways
- Virtual Private Gateways
- VPN Connections

Buttons: [Create Internet Gateway](#) [Delete](#) [Attach to VPC](#) [Detach from VPC](#)

Search Internet Gateways and

<< 1 to 4 of 4 Internet Gateways >>

| <input type="checkbox"/> | Name | ID | State | VPC |
|-------------------------------------|----------------------|--------------|----------|------------------------------------|
| <input type="checkbox"/> | Gateways TP3 Nesello | igw-19f31b7e | attached | vpc-9ef4def9 Rede TP3 |
| <input checked="" type="checkbox"/> | GW-Lucas-Projeto | igw-5fcb6238 | attached | vpc-32699a54 VPC-Lucas-Projeto |
| <input type="checkbox"/> | Public GW - AT | igw-a9bf1ece | attached | vpc-c954bc af Rodrigo-VPC_AT |
| <input type="checkbox"/> | GW-Lucas-Alonso-AT | igw-f075db97 | attached | vpc-3d8c7f5b VPC-Lucas-Alonso-AT |

igw-5fcb6238 | GW-Lucas-Projeto

Summary Tags

ID: igw-5fcb6238 | GW-Lucas-Projeto

Attached VPC ID: vpc-32699a54 | VPC-Lucas-Projeto

State: attached

Attachment state: available

Figura 5. Criação do gateway de rede GW-Lucas-Projeto

Em seguida configuramos as rotas do gateway. Nesse caso só inserimos uma rota padrão para o endereço do gateway de forma que os clientes na sub-rede possam acessar a internet.

VPC Dashboard

Filter by VPC: None

Virtual Private Cloud

- Your VPCs
- Subnets
- Route Tables**
- Internet Gateways
- Egress Only Internet Gateways
- DHCP Options Sets
- Elastic IPs
- Endpoints
- NAT Gateways
- Peering Connections
- Security
- Network ACLs
- Security Groups
- VPN Connections
- Customer Gateways
- Virtual Private Gateways
- VPN Connections

Buttons: [Create Route Table](#) [Delete Route Table](#) [Set As Main Table](#)

Search Route Tables and their

<< 1 to 5 of 5 Route Tables >>

| <input type="checkbox"/> | Name | Route Table ID | Explicitly Associat | Main | VPC |
|-------------------------------------|-----------------------|----------------|---------------------|------|------------------------------------|
| <input type="checkbox"/> | | rtb-d05c23b6 | 0 Subnets | Yes | vpc-32699a54 VPC-Lucas-Projeto |
| <input type="checkbox"/> | | rtb-adde04cb | 0 Subnets | Yes | vpc-9ef4def9 Rede TP3 |
| <input type="checkbox"/> | | rtb-03c5ae65 | 2 Subnets | Yes | vpc-c954bc af Rodrigo-VPC_AT |
| <input type="checkbox"/> | Tabela de Rotas - VPC | rtb-bdabd7db | 2 Subnets | Yes | vpc-3d8c7f5b VPC-Lucas-Alonso-AT |
| <input checked="" type="checkbox"/> | Routes-Lucas-Projeto | rtb-ae611ec8 | 1 Subnet | No | vpc-32699a54 VPC-Lucas-Projeto |

rtb-ae611ec8 | Routes-Lucas-Projeto

Summary Routes Subnet Associations Route Propagation Tags

Edit

View: All rules

| Destination | Target | Status | Propagated |
|---------------|--------------|--------|------------|
| 172.16.0.0/16 | local | Active | No |
| 0.0.0.0/0 | igw-5fcb6238 | Active | No |

Figura 6. Configuração da tabela de rotas

Por fim, configuramos um grupo de segurança que permite o acesso HTTP, HTTPS, SSH, e MySQL (somente internamente). Ele age como um firewall e bloqueia tudo que não for liberado para entrar na nossa rede.

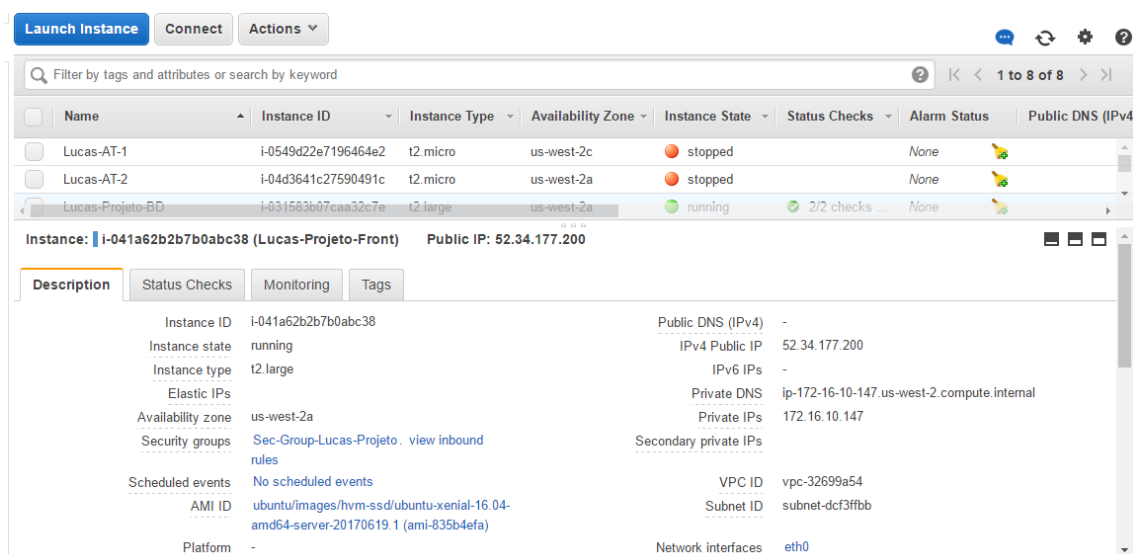


Figura 7. Configuração Grupo de Segurança

Com estas configurações realizadas já possuímos uma infraestrutura de rede funcional na Amazon AWS. Esperando nossas instancias EC2.

5.5.1.2. Instancias EC2

Para criar as instancias necessárias, acessamos os serviços da AWS e selecionamos EC2. No painel de gerenciamento web, clicamos em lançar instancia. Logo somos direcionados para o instalador da instancia. Onde definimos os requisitos a cada passo que avançamos.

Primeiro definimos a imagem de sistema operacional a ser usada. Em nosso caso, escolhemos a imagem do ubuntu 16.04 LTS.

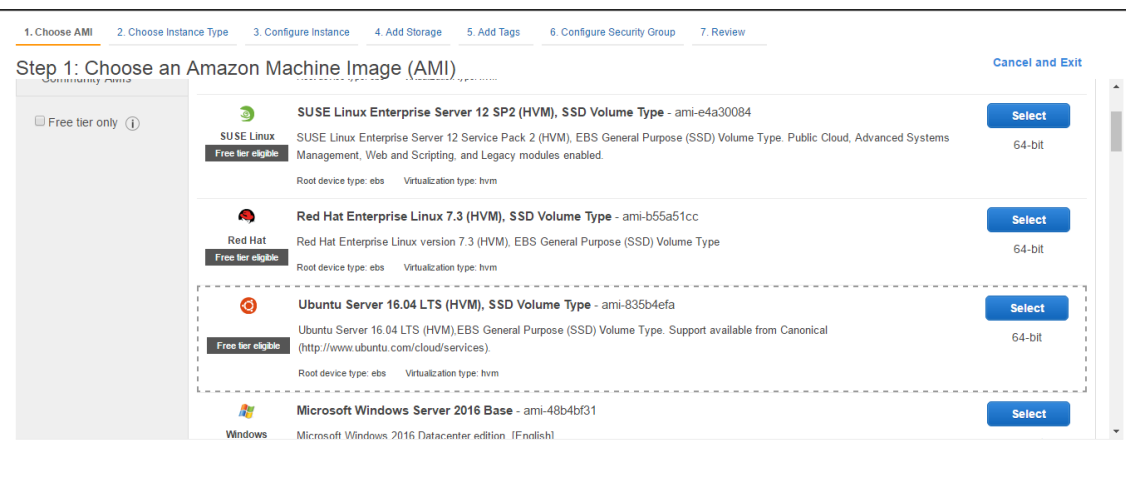


Figura 8. Seleção de imagem utilizada

Em seguida, definimos o tipo da instancia que desejamos criar de acordo com os requisitos estipulados para memória e CPU. Escolhemos a versão t2.medium.

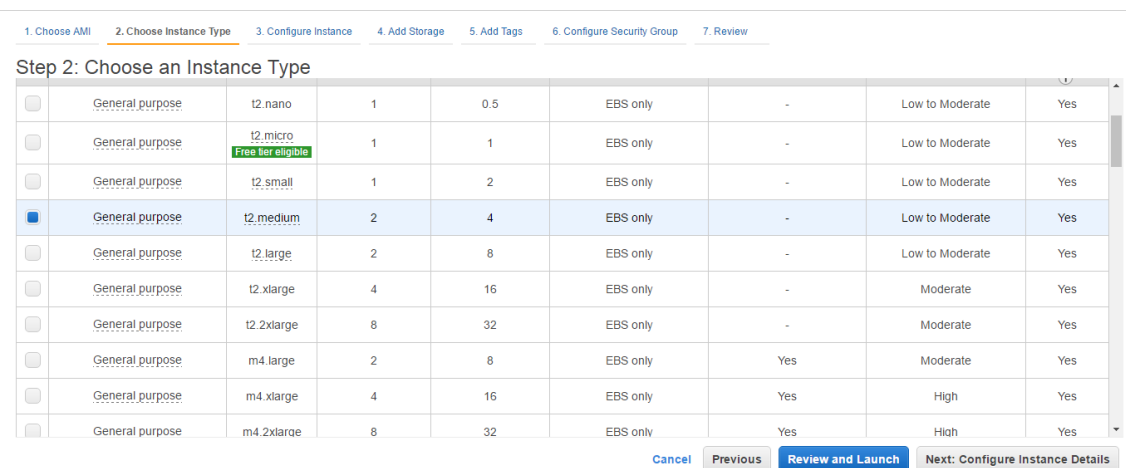


Figura 9. Seleção do tipo de instancia

Após a definições de memória e processamento, definimos as configurações de rede da instancia. A que VPC irá se conectar, a qual sub-rede, etc.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances [Launch into Auto Scaling Group](#)

Purchasing option ☐ Request Spot instances

Network [Create new VPC](#)

Subnet [Create new subnet](#)

Auto-assign Public IP

IAM role [Create new IAM role](#)

⚠ You do not have permissions to list any IAM roles. Contact your administrator, or check your IAM permissions.

Shutdown behavior

Enable termination protection ☐ Protect against accidental termination

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Storage](#)

Figura 10. Definições de rede da instancia

Agora definimos o tamanho da storage usada. Novamente seguindo os requisitos estipulados no projeto, setamos a variável para 500GB.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

| Volume Type | Device | Snapshot | Size (GiB) | Volume Type | IOPS | Throughput (MB/s) | Delete on Termination | Encrypted |
|-------------|-----------|------------------------|----------------------------------|---------------------------|-------------|-------------------|-------------------------------------|---------------|
| Root | /dev/sda1 | snap-039ffe051a6c30204 | <input type="text" value="500"/> | General Purpose SSD (GP2) | 1500 / 3000 | N/A | <input checked="" type="checkbox"/> | Not Encrypted |

[Add New Volume](#)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Tags](#)

Figura 11. Definição de storage da instancia

Por fim, configuramos o grupo de segurança a ser utilizado e lançamos a instancia.

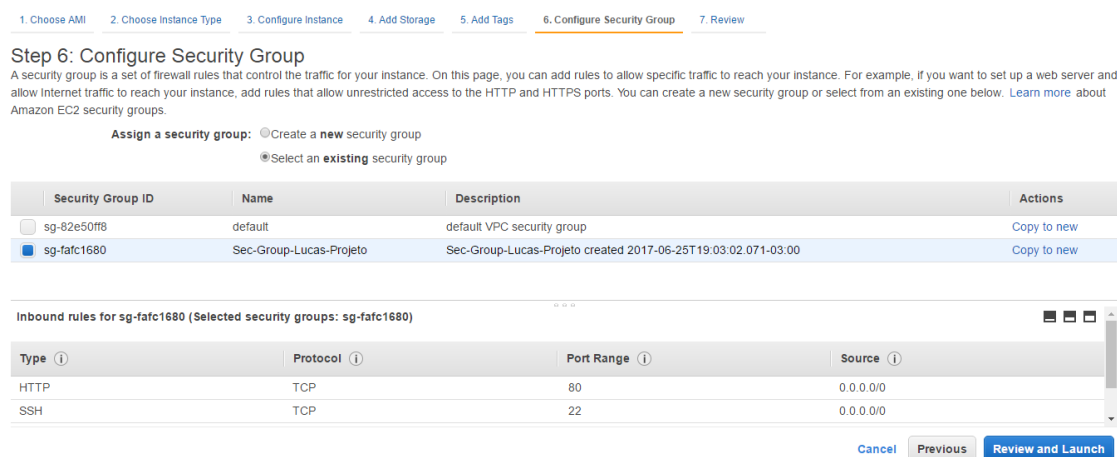


Figura 12. Definição do grupo de segurança utilizado pela instancia

Assim que lançamos a instancia, voltamos ao painel principal das instancias EC2 e podemos vê-la rodando. Executamos os mesmos processos para criar ambas as máquinas e demos os nomes Lucas-Projeto-Front e Lucas-Projeto-BD para o webserver e o servidor de banco de dados, respectivamente.

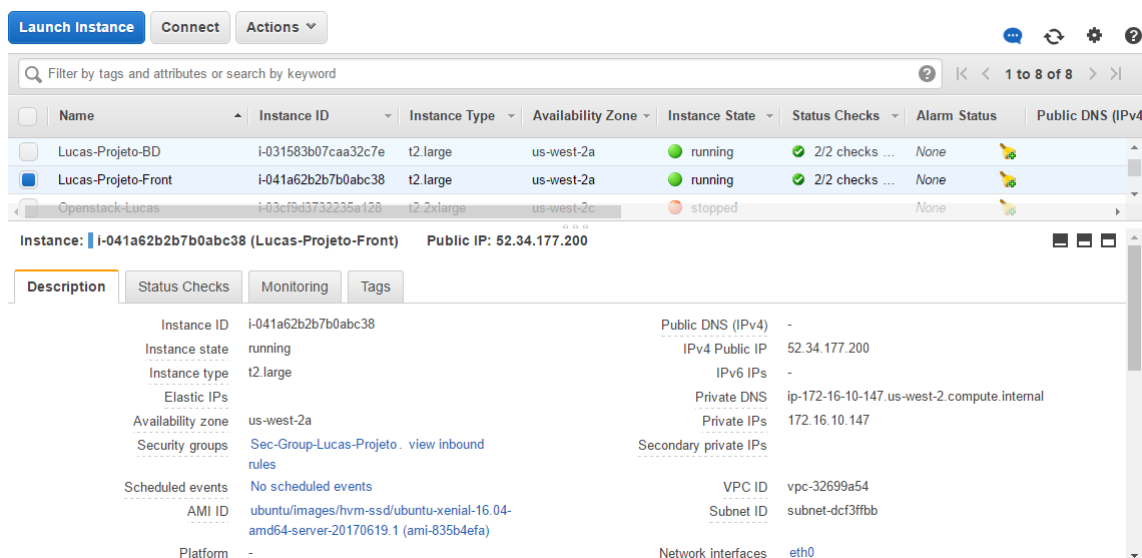


Figura 13. Instancia Webserver lançada

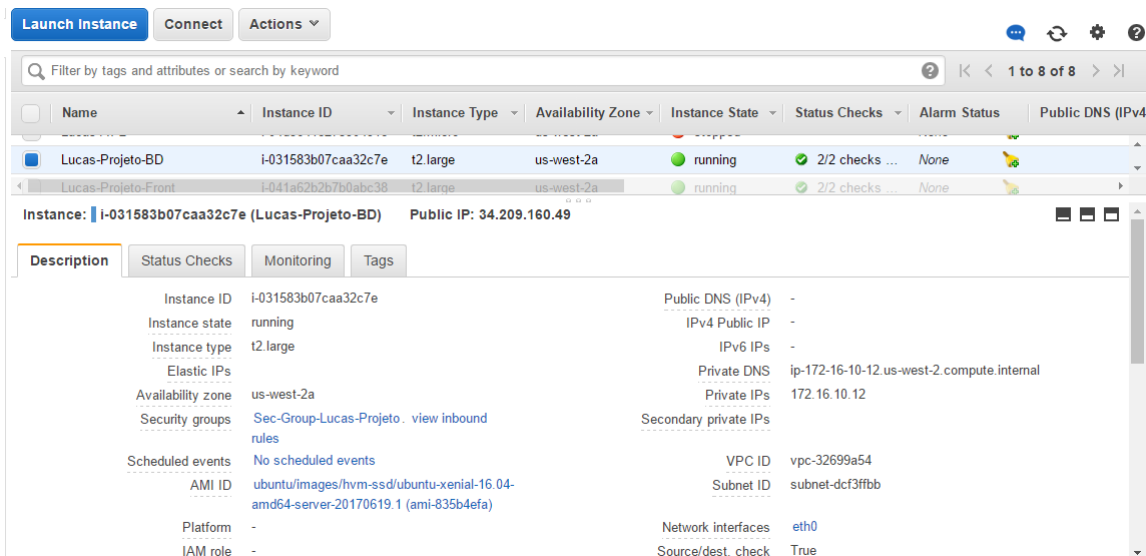


Figura 14. Instancia banco de dados lançada

Com as instancias iniciadas, agora devemos acessa-las e rodar o ansible para instalar a aplicação Wordpress.

5.5.2. Aplicação

O wordpress é uma aplicação simples de ser instalada. Como vimos nos requisitos do sistema, o wordpress roda em cima de bibliotecas PHP, em um servidor web apache com o banco de dados MySQL. Com uma uma simples instalação da pilha LAMP e algumas configurações adicionais podemos instala-lo de maneira totalmente funcional.

Pensando a longo prazo, podemos precisar reinstalar este sistema diversas vezes, seja para teste, como serviço prestado, cluster, etc. Sendo assim, vamos utilizar uma ferramenta de gerenciamento da configuração para automatizar a instalação do software e todas suas dependencias, assim como as configurações posteriores necessárias, o Ansible.

O Ansible é uma feramenta open-source que utiliza scripts em YAML para executar tarefas previamente programadas. Com isso, podemos desenvolver um código para a instalação do wordpress e instala-lo com apenas uma linha

de comando a partir da ferramenta ansible-playbook que vem com o pacote ansible em sua instalação.

5.5.2.1. Ansible

Para instalar o ansible precisamos adicionar seu repositório de instalação no sistema com o comando:

```
$ sudo apt-add-repository ppa:ansible/ansible
```

Em seguida atualizamos a lista de repositórios e instalamos o ansible com o gerenciador de pacotes padrão do ubuntu.

```
$ sudo apt-get update  
$ sudo apt-get install ansible
```

Com a ferramenta instalada, criamos um diretório para armazenar nosso código chamado “distributed-wordpress”.

```
$ sudo mkdir distributed-wordpress  
$ cd distributed-wordpress
```

Dessa forma acessamos o diretório criado e podemos dar início a produção de nosso código.

O ansible é executado a partir de playbooks. Os playbooks ordenam as tarefas que serão executadas pelos scripts YAML. Sendo assim, criamos nosso arquivo chamado playbook.yml e inserimos os seguintes dados.


```
- hosts: frontend
```

```
roles:
```

```
- server
```

```
- php
```

```
- wordpress
```

```
- hosts: backend
```

```
roles:
```

```
- server
```

```
- mysql
```

Tabela 1. Arquivo playbook.yml

Como podemos ver, em nosso playbook definimos as roles que serão executadas. As roles serão criadas mais para frente, porém já as nomeamos agora e identificamos o nome do grupo de hosts que sofrerão alterações por cada role. Para isso também criamos um arquivo chamado hosts que define os grupos de máquinas que sofrerão mudanças.

```
[frontend]
```

```
172.16.10.147 ansible_ssh_private_key_file=/Projeto-Lucas-2606.pem
```

```
[backend]
```

```
172.16.10.12      ansible_ssh_private_key_file=/Projeto-Lucas-2606.pem
ansible_python_interpreter=/usr/bin/python3
```

Tabela 2. Arquivo hosts

Neste arquivo criamos os grupos de máquinas backend e frontend, em cada grupo inserimos os IP's das instancias EC2 criadas em seu devido local. Banco de dados como backend e servidor web como frontend. Também inserimos o caminho a chave SSH privada que foi gerada na criação das instancias. O ansible depende da chave para ser executado via ssh em servidores remotos. Por fim forçamos a execução do interpretador python mais novo para a execução dos scripts python.

Com os arquivos hosts e playbook devidamente criados, passamos para as roles que serão executadas. Criamos um diretório chamado “/distributed-wordpress/roles/”.

Como vimos no arquivo playbook, dividimos as roles em “server”, “php”, “wordpress” e “mysql”. As três primeiras serão executadas no servidor web, enquanto a ultima somente no servidor de banco de dados.

As roles possuem uma estrutura de diretórios singular. Por isso o ansible possui uma ferramenta de criação desta estrutura chamada ansible-galaxy. Sendo assim, a utilizamos e criamos a estrutura necessária para cada role com os comandos a seguir.

```
$ sudo ansible-galaxy init server
$ sudo ansible-galaxy init php
$ sudo ansible-galaxy init mysql
$ sudo ansible-galaxy init wordpress
```

Tabela 3. Execução ansible-galaxy

Com esse comando, foram criados diretórios com sub-diretórios próprios para a execução do ansible.

```
$ ls -l
total 28
drwxr-xr-x 2 root root 4096 Mar 26 00:15 defaults
drwxr-xr-x 2 root root 4096 Mar 26 00:15 handlers
drwxr-xr-x 2 root root 4096 Mar 26 00:15 meta
-rw-r--r-- 1 root root 1328 Mar 26 00:15 README.md
drwxr-xr-x 2 root root 4096 Mar 26 00:15 tasks
drwxr-xr-x 2 root root 4096 Mar 26 00:15 tests
drwxr-xr-x 2 root root 4096 Mar 26 00:15 vars
```

Tabela 4. Estrutura de diretórios das roles

Com essa estrutura criada, basta inserirmos os comandos nos diretórios corretos. Começaremos pela role do banco de dados. Nela precisamos inserir as variáveis do banco de dados, que serão utilizadas ao longo do projeto, como o nome do usuário do banco, o nome do banco, e a senha do usuário. Para isto editamos o arquivo:

distributed-wordpress/roles/mysql/defaults/main.yml

```
---

wp_mysql_db: wordpress
wp_mysql_user: wordpressuser
wp_mysql_password: wordpresspassword

# defaults file for mysql
```

Tabela 5. Variáveis MySQL

Dessa forma as variáveis serão reconhecidas durante a execução do código.

Em seguida editamos o arquivo “distributed-wordpress/roles/mysql/handlers/main.yml”. Neste criamos um handlers, como dis o diretório, para reiniciar o servidor MySQL. Dessa forma, basta chamarmos o nome definido neste arquivo e o comando de reinicio do serviço será executado assim que chamado.

```
---  
  
- name: Reiniciando o MySQL  
  service: name=mysql state=restarted  
  
  become: yes  
  
# handlers file for wordpress
```

Tabela 6. Handler MySQL

Por fim, definimos as tarefas que serão executadas de fato nesta role, no arquivo “distributed-wordpress/roles/mysql/tasks/main.yml”.

```
---  
  
- name: Update apt cache  
  apt: update_cache=yes cache_valid_time=3600  
  
  become: yes
```

```

- name: Install required software

  apt: name={{ item }} state=present

  become: yes

  with_items:

    - python-mysqldb

    - mysql-server

    - php-mysql

    - python-mysqldb-dbg

    - python-software-properties

- name: Atualizar arquivo de configuracao do wordpress

  become: yes

  lineinfile:

    dest=/etc/mysql/mysql.conf.d/mysqld.cnf      regexp='^bind-address      =
127\.\0\.\0\.\1' line='bind-address = 0.0.0.0'

  notify:

    - Reiniciando o MySQL

- name: Create mysql database

  become: yes

  mysql_db:

    name={{ wp_mysql_db }}      state=present      login_user=root
login_password=SenhaRoot

- name: Create mysql user

```

```

become: yes

mysql_user: name={{ wp_mysql_user }} password={{ wp_mysql_password }}
priv='*.*:ALL,GRANT' state=present login_user=root
login_password=SenhaRoot host=172.16.10.147

# tasks file for mysql

```

Tabela 7. Tarefas MySQL

Finalizamos o código para o banco de dados. Agora seguiremos adiante com as demais roles. As configurações de cada role seguem o mesmo padrão, porém nem sempre será necessário configurar os arquivos defaults ou handlers. Por exemplo, para a role PHP só configuramos as tarefas que serão executadas no diretório tasks. Pois só ela bastaria para a instalação e configuração do PHP. Sendo assim editamos o arquivo:

“distributed-wordpress/roles/mysql/tasks/main.yml”

```

---

- name: Install PHP extensions

  apt: name={{ item }} state=present

  sudo: yes

  with_items:

    - php-gd

    - python-mysqldb

# tasks file for php

```

Tabela 8. Tarefas PHP

Daremos continuidade com a role server. Esta role também só precisamos configurar as tarefas executadas e nada mais. Por isso só editamos o arquivo:

distributed-wordpress/roles/mysql/tasks/main.yml, como podemos ver abaixo.

```
---  
  
- name: Update apt cache  
  apt: update_cache=yes cache_valid_time=3600  
  sudo: yes  
  
- name: Install required software  
  apt: name={{ item }} state=present  
  sudo: yes  
  with_items:  
    - apache2  
    - php-mysql  
    - php  
    - libapache2-mod-php  
    - php-mcrypt  
    - python3-mysqldb  
    - python-pip  
  
# tasks file for server
```

Tabela 9. Tarefas para servidor

Por fim, configuramos a role wordpress. Nesta role também configuraremos as variáveis do banco de dados, pois durante sua execução, vamos atualizar o arquivo de configuração do wordpress para utilizar essas variáveis. Sendo assim, configuramos o arquivo:

distributed-wordpress/roles/wordpress/defaults/main.yml

```
---

wp_mysql_db: wordpress
wp_mysql_user: wordpressuser
wp_mysql_password: wordpresspassword

# defaults file for wordpress
```

Tabela 10. Variáveis wordpress

Como esta é uma role que mexemos bastante em seus arquivos de configuração, também configuramos um Handler para executar um reinício do serviço web do servidor, que foi instalado durante a execução da role “server”. Para isso editamos o arquivo:

distributed-wordpress/roles/wordpress/handlers/main.yml

```
---

- name: Reiniciando o Apache

  service: name=apache2 state=restarted

  sudo: yes

# handlers file for wordpress
```

Tabela 11. Handler wordpress

Finalizando o código ansible, configuramos as tarefas executadas na role wordpress. Configuramos o download e extração do projeto, atualizamos o diretório raiz do apache para servir o wordpress, configuramos o arquivo de configuração “wp-config”, etc. Para isso, editamos o arquivo:


```
---

- name: Download wordpress
  get_url:
    url=https://wordpress.org/latest.tar.gz
    dest=/tmp/wordpress.tar.gz
    validate_certs=no
  sudo: yes

- name: Extracao do wordpress para diretorio do apache
  unarchive:
    src=/tmp/wordpress.tar.gz
    dest=/var/www/
    copy=no
  sudo: yes

- name: Atualizar documento raiz para o wordpress
  sudo: yes
  lineinfile:
    dest=/etc/apache2/sites-enabled/000-default.conf
    regexp="(.)+DocumentRoot      /var/www/html"      line="DocumentRoot
/var/www/wordpress"
  notify:
    - Reiniciando o Apache
```

```

- name: Configurações do Wordpress

  command:

    mv /var/www/wordpress/wp-config-sample.php /var/www/wordpress/wp-
    config.php creates=/var/www/wordpress/wp-config.php

  sudo: yes


- name: Atualizar arquivo de configuracao do wordpress

  sudo: yes

  lineinfile:

    dest=/var/www/wordpress/wp-config.php regexp="{{ item.regexp }}" line="{{
    item.line }}"

  with_items:

    - {'regex': "define\\('DB_NAME', '(.)+'\\);", 'line': "define('DB_NAME', '{{
    wp_mysql_db }}');"}

    - {'regex': "define\\('DB_USER', '(.)+'\\);", 'line': "define('DB_USER', '{{
    wp_mysql_user }}');"}

    - {'regex': "define\\('DB_PASSWORD', '(.)+'\\);", 'line': "define('DB_PASSWORD', '{{
    wp_mysql_password }}');"}

# tasks file for wordpress

```

Tabela 12. Tarefas role wordpress

Ao final das configurações de cada role, do arquivo hosts e do arquivo playbook, chegou a hora de executarmos o código e vermos se está tudo ok. Para isso, utilizamos a ferramenta de execução de playbooks do ansible “ansible-playbook”. Especificamos o arquivo hosts que será utilizado e utilizamos o usuário com privilégios para fazê-lo.

```
:~/distributed-wordpress$ sudo ansible-playbook playbook.yml -i hosts -u ubuntu
```

```
sudo: unable to resolve host ip-172-16-10-147
```

```
[DEPRECATION WARNING]: Instead of sudo/sudo_user, use become/become_user and
```

```
make sure become_method is 'sudo' (default).
```

```
This feature will be removed in a
```

```
future release. Deprecation warnings can be disabled by setting
```

```
deprecation_warnings=False in ansible.cfg.
```

```
PLAY [frontend] *****
```

```
TASK [Gathering Facts] *****
```

```
ok: [172.16.10.147]
```

```
TASK [server : Update apt cache] *****
```

```
ok: [172.16.10.147]
```

```
TASK [server : Install required software] *****
```

```
ok: [172.16.10.147] => (item=[u'apache2', u'php-mysql', u'php', u'libapache2-
```

```
mod-php', u'php-mcrypt', u'python3-mysqldb', u'python-pip'])
```

```
TASK [php : Install PHP extensions] *****
```

```
ok: [172.16.10.147] => (item=[u'php-gd', u'python-mysqldb'])
```

```
TASK [wordpress : Download wordpress] *****
```

```
changed: [172.16.10.147]
```

```
TASK [wordpress : Extracao do wordpress para diretorio do apache]
```

```
*****
```

```
ok: [172.16.10.147]
```

```
TASK [wordpress : Atualizar documento raiz para o wordpress]
```

```
*****
```

```
changed: [172.16.10.147]
```

```
TASK [wordpress : Configurações do Wordpress]
```

```
*****
```

```
ok: [172.16.10.147]
```

```
TASK [wordpress : Atualizar arquivo de configuracao do wordpress]
```

```
*****
```

```
ok: [172.16.10.147] => (item={u'regexp': u"define\\('DB_NAME', '(.)+'\\);",  
u'line': u"define('DB_NAME', 'wordpress');"})
```

```
ok: [172.16.10.147] => (item={u'regexp': u"define\\('DB_USER', '(.)+'\\);", u'line':  
u"define('DB_USER', 'wordpressuser');"})
```

```
ok: [172.16.10.147] => (item={u'regexp': u"define\\('DB_PASSWORD', '(.)+'\\);",  
u'line': u"define('DB_PASSWORD', 'wordpresspassword');"})
```

```
RUNNING HANDLER [wordpress : Reiniciando o Apache]  
*****
```

```
changed: [172.16.10.147]
```

```
PLAY [backend] *****
```

```
TASK [Gathering Facts] *****
```

```
ok: [172.16.10.12]
```

```
TASK [server : Update apt cache] *****
```

```
ok: [172.16.10.12]
```

```
TASK [server : Install required software] *****
```

```
ok: [172.16.10.12] => (item=[u'apache2', u'php-mysql', u'php', u'libapache2-  
mod-php', u'php-mcrypt', u'python3-mysqldb', u'python-pip'])
```

```
TASK [mysql : Update apt cache] *****
```

```
ok: [172.16.10.12]
```

```
TASK [mysql : Install required software] *****
```

```
ok: [172.16.10.12] => (item=[u'python-mysqldb', u'mysql-server', u'php-mysql',  
u'python-mysqldb-dbg', u'python-software-properties'])
```

```

TASK [mysql : Atualizar arquivo de configuracao do wordpress]
*****

changed: [172.16.10.12]

TASK [mysql : Create mysql database] *****
ok: [172.16.10.12]

TASK [mysql : Create mysql user] *****
ok: [172.16.10.12]

RUNNING HANDLER [mysql : Reiniciando o MySQL]
*****

changed: [172.16.10.12]

PLAY RECAP *****
172.16.10.12      : ok=9  changed=2  unreachable=0  failed=0
172.16.10.147    : ok=10 changed=3  unreachable=0  failed=0

```

Tabela 13. Execução ansible

Ao final deste comando, podemos ver que o ansible foi executado com sucesso em ambos os servidores. Tivemos 2 mudanças no servidor de banco de dados e 3 no servidor web. Todas as tarefas executadas tiveram saída “OK” e não houve nenhuma falha durante a execução. Sendo assim, podemos concluir que a aplicação foi instalada com sucesso e já está rodando em nosso servidor web utilizando o banco de dados de outro servidor, de forma distribuída.

Para conferirmos a instalação, acessamos a instancia EC2 pelo IP público designado pela amazon.

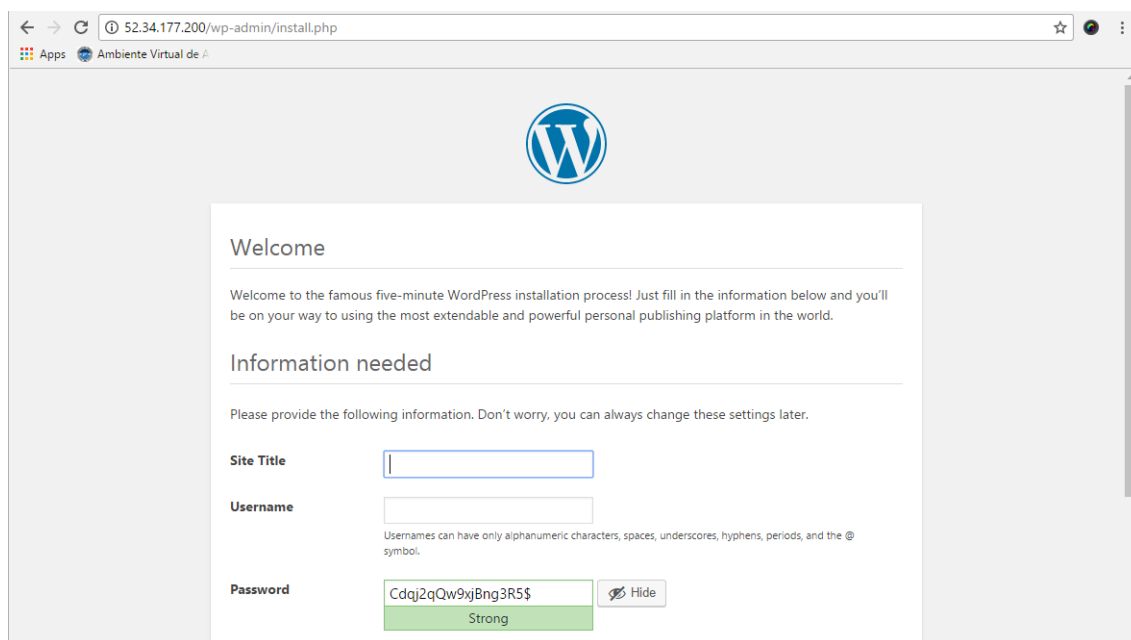


Figura 15. Wordpress funcionando

Como podemos ver, fomos direcionados para a página de instalação do wordpress, o que prova a instalação correta da aplicação de forma distribuida, em um ambiente de nuvem privada fornecido pela amazon AWS.

Objetivando o fácil acesso ao código, vamos utilizar o sistema de versionamento GIT para hospedar nosso código, que pode ser acessado no endereço: <https://github.com/alonsolucas/distributed-wordpress.git>

6. Conclusões

Concluimos nosso projeto com a implementação bem sucedida de um web site baseado em wordpress, com a aplicação distribuida em duas instancias EC2 da Amazon AWS. Também utilizamos o serviço Virtual Private Cloud para criar um ambiente de nuvem privada para hospedar nossa aplicação.

Podemos entender que o prazo e os recursos planejados para este projeto, como capacidade de processamento, quantidade de memória RAM e tamanho de storage, foram mais do que suficientes. Inclusive, nos primeiros meses de utilização, recomendaria uma redução do tipo das instancias usadas.

Com o sucesso deste projeto inicial, podemos atingir os objetivos da diretoria futuramente. Migrar o datacenter atual para a nuvem. A próxima melhoria a fazer será utilizar um load balancer, com o serviço ELB, para nosso web site e criar uma nova sub-rede em outra zona de disponibilidade da amazon, de forma a garantir alta disponibilidade deste serviço.

Sendo assim, concluimos nosso projeto de forma satisfatória. Todos os serviços foram implementados com sucesso, inclusive um código YAML para o ansible. Todos os arquivos foram exportados para o github para que possam ser acessados e clonados a qualquer momento, de qualquer lugar, além de possibilitar a interação com outros colaboradores e ser um ótimo sistema de versionamento de códigos.

7. Referências

<https://www.vmware.com/br/solutions/virtualization.html>

<https://docs.vmware.com/en/VMware-vSphere/index.html>

<https://cloud.google.com/?hl=pt-br>

<https://cloud.google.com/compute/>

<https://www.docker.com/what-docker>

<https://www.openstack.org/>

<https://aws.amazon.com/pt/what-is-aws/>

<https://aws.amazon.com/pt/solutions/>

<https://calculator.s3.amazonaws.com/index.html>

<https://pt.wikipedia.org/wiki/WordPress>

<https://code.trac.wordpress.org/browser>

<git://core.git.wordpress.org/>

<https://wordpress.org/download/source/>

<https://pt.wikipedia.org/wiki/WordPress>

[https://codex.wordpress.org/pt-br:Requisitos do WordPress](https://codex.wordpress.org/pt-br:Requisitos_do_WordPress)

<http://calculator.s3.amazonaws.com/index.html#r=IAD&key=calc-6AFD5D5B-218A-43E5-9BD7-1238B0F07EDA>

<https://github.com/alonsolucas/distributed-wordpress.git>