

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



**ARQUITETURA E INFRAESTRUTURA DE**  
**APLICAÇÕES**

**ALUNO: Rodrigo Nesello Figueiredo – Matr. 7612**

**E-MAIL: [rodrigo.figueiredo@al.infnet.edu.br](mailto:rodrigo.figueiredo@al.infnet.edu.br)**

**TURMA: GRC – manhã**

**Projeto de Bloco - Arquitetura e Infraestrutura de**  
**Aplicações**  
**Projeto Final**

## Sumário

1. Introdução.....	3
2. Objetivo .....	3
3. Dados do Projeto .....	3
4. Datacenter.....	4
5. Infraestrutura .....	4
6. Funcionamento da Infraestrutura .....	6
7. Implementação .....	7
8. Solução de virtualização.....	8
9. Outras soluções de virtualização .....	8
10. Referência de downloads.....	9
11. Infraestrutura de virtualização .....	9
12. Planejamento de instalação.....	10
13. Documentando repositório .....	15
14. Cirando o repositório .....	16
15. Aplicação distribuída com dois containers Docker .....	20
16. Automatizando instalação de Containers .....	26
17. Cronograma de instalação .....	28
18. Conclusão .....	28
19. Referências .....	29

## **1. Introdução**

Este projeto tem como premissa escolher uma aplicação que atenda às necessidades e o modelo de negócios do OGMORJ. Sua implantação será em um ambiente com infraestrutura de nuvem privada.

Será descrito no projeto a arquitetura de virtualização a ser implementada, comparando com outras abordagens de hypervisor's, bem como um playbook Ansible que realizará a instalação da aplicação distribuída WordPress de forma automatizada.

Mostraremos referências dos downloads e/ou versões de todos os componentes do WordPress, bem como seus passos de implementação de sua infraestrutura de virtualização.

## **2. Objetivo**

Apresentar de forma organizada e clara as características de desenvolvimento, infraestrutura, implantação e configuração da aplicação escolhida, bem como seus detalhes técnicos.

Apresentar os passos de uma instalação automatizada do WordPress através do Ansible, seu cronograma com estimativa de tempo para execução de cada atividade e os passos de configuração e funcionamento da aplicação.

## **3. Dados do Projeto**

Tratasse de uma empresa do ramo de gestão de mão-de-obra portuária, responsável pela gestão nos portos do Rio de Janeiro, Itaguaí, Niterói e Forno, conforme descrito na Lei Nº 8630, de 25 de fevereiro de 1993, Art. 18, que determina aos operadores portuários a constituir, em cada porto organizado, um órgão gestor de mão-de-obra portuária avulsa. As operadoras portuárias são obrigadas por força de Lei Federal a requisitar mão de obra portuária avulsa através do OGMO e para isso escolhemos adotar o WordPress.

Como o OGMO precisa ofertar vagas de trabalho requisitadas pelas operadoras portuárias, resolvemos adotar o WordPress para recebermos e centralizarmos estas requisições de forma que fique fácil o seu gerenciamento.

O WordPress é uma ferramenta Open-Source, ou seja, sem nenhum custo de licenciamento, e distribuída sob a licença GPLv2, e isso faz dela uma aquisição rápida e simples de se implementar.

O WordPress faz o gerenciamento e publicação de conteúdo para web, com banco de dados MySQL, escrito em PHP, com foco em criação de sites e blogs via web. Ele também possui utilidades como a instalação de plugins como o “All-in-one Intranet” e o “Intranet Plus” permitindo a implementação de uma intranet.

O WordPress é desenvolvido em PHP; possui Backend com banco de dados MySQL; e Front-end com servidor WEB Apache.

Seu desenvolvimento apesar de ser Open-Source está associada a empresa Automattic, onde controla o versionamento tendo a versão estável mais atual da

aplicação sendo a 4.7.2 podendo ser encontrada no link [https://br.wordpress.org/latest-pt\\_BR.zip](https://br.wordpress.org/latest-pt_BR.zip).

O WordPress possui como seus principais recursos o de Gerar XML, XHTML, e CSS em conformidade com os padrões W3C; O gerenciamento integrado de ligações; possui 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 a partir da versão 2.3; Gerencia múltiplos blogs em subpastas ou subdomínios desde a versão 3.0; Importa e exporta dados; API de desenvolvimento de plug-ins; Possui níveis, promoção e rebaixamento de usuários; E campos personalizados para armazenamento de dados extras no banco de dados.

Conforme o site da comunidade wordpress, seus requisitos de sistema para instalação são: Servidor baseado em UNIX/Linux1, PHP versão 5.2.4 ou superior, MySQL versão 5.0 ou superior, Memória para o PHP de pelo menos 64 MB (Somente para o software WordPress, sem plugins adicionais).

Como recursos extras ele precisa de: 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).

Os clientes terão acesso aos recursos de criação de requisição de trabalho através do Wordpress por meio de qualquer navegador a partir de suas máquinas cliente.

#### **4. Datacenter**

A fim de automatizar o ambiente, será implementado um datacenter do tipo SDDC (Software Defined Data Center), baseada em VMware, com dois servidores virtualizados, sendo um servidor de banco de dados MySQL e um Apache web server.

#### **5. Infraestrutura**

O Wordpress possui infraestrutura de implantação baseada em cliente servidor onde o cliente faz requisições via web através de browser para o servidor frontend, que repassa esta requisição para o servidor de banco de dados MySQL que por sua vez devolve a informação para o servidor apache. Ocorrido isto o Apache entrega ao cliente a requisição fechando o ciclo de comunicação.

A infraestrutura necessária para atender este projeto deverá conter:

➤ Roteador de Borda:  
Mikrotik Router Board 750r2 Hex lite  
CPU: 850 MHz de 1 núcleo  
Memória: 64 MB RAM  
Arquitetura: MIPS-BE  
Rede: 5 Portas Ethernet 10/100  
Sistema operacional: RouterOS

➤ SrvFW (FireWall)

HP ProLiant DL380 Gen9 Server

Processador: Intel Xeon CPU E5 - 2650 v3 @2.30 GHz 2.30 GHz (2 Processadores)

Memória: 8 GB RAM

Disco: 200 GB SATA

Rede: 1GbE 4Port Ethernet Adapter

SO: Untangle 12.0

➤ SrvHV (Host de Virtualização)

HP ProLiant DL380 Gen9 Server

Processador: Intel Xeon CPU E5 - 2650 v3 @2.30 GHz 2.30 GHz (2 Processadores)

Memória: 32 GB RAM

Disco: 500 GB SAS

Rede: 1GbE 4Port Ethernet Adapter

SO: VMWare ESXi

➤ Maquinas virtuais

=> **SrvWB** (Servidor Web) – Servidor Linux Ubuntu Server com Apache WEB Server e PHP na última versão estável, 8GB RAM, 120GB HD, 2 processadores de 2 núcleos.

=> **SrvBD** – (Servidor Banco de Dados) – Servidor Linux Ubuntu Server com banco de dados MySQL na última versão estável, 8GB RAM, 120GB HD, 2 processadores de 2 núcleos.

➤ Switch Hp 48p E2510-48 J9020a (4 Switch)

Portas: 48 portas 10/100/1000 RJ-45 com negociação automática e 4 portas SFP 1000 Mbps

Memória e processador: MIPS a 650 MHz e 32 MB de flash

Tamanho do buffer de pacotes: 12 Mb e SDRAM de 128 MB

➤ Storage STC-R2600

Capacidade de Discos: 24 BAIAS 4U

Processador: Intel Xeon E5-2620v4 Octa Core 2.1 Ghz TURBO: 3GHZ - Cache: 20MB

MEMÓRIA: 1 x 8GB - DDR4-2133MHZ RDIMM ECC

Tipo de disco: SATA ou SAS 3,5 - Hot Swap

Nível de RAID: 0, 1, 5, 6, 10, 50, 60 - Via hardware

Controladora: AOC-S3108L-H8IR SAS 3.0 Chipset: LSI SAS 3108

REDE: 4x Intel i350 Gigabit Ethernet (10/100/1000 Mbit) e 1x IPMI

➤ UPS

2 (dois) Nobreak 3Kva Apc Smart-Ups Senoidal Mono 230V Sua3000I

➤ Rack

Rack Piso fechado Padrão 44UX80cmX110cm – Preto – Todo Perfurado

## 6. Funcionamento da Infraestrutura

Toda infraestrutura citada no item 5 deste documento, funcionará de forma a atender as necessidades do projeto obedecendo as seguintes normas de execução:

O roteador de borda da Mikrotik receberá o link de acesso à internet, que retransmitirá o sinal para o nosso FireWall Untangle 12.0 (SrvFW) onde este será encarregado de realizar os devidos filtros e proteções antes de encaminhar o acesso para o resto da rede.

Logo após o acesso à internet ter sido tratado pelo firewall, o mesmo encaminhará para os 4 Switch que formam redundância de rede, ocorrendo a redistribuição do acesso para toda a rede.

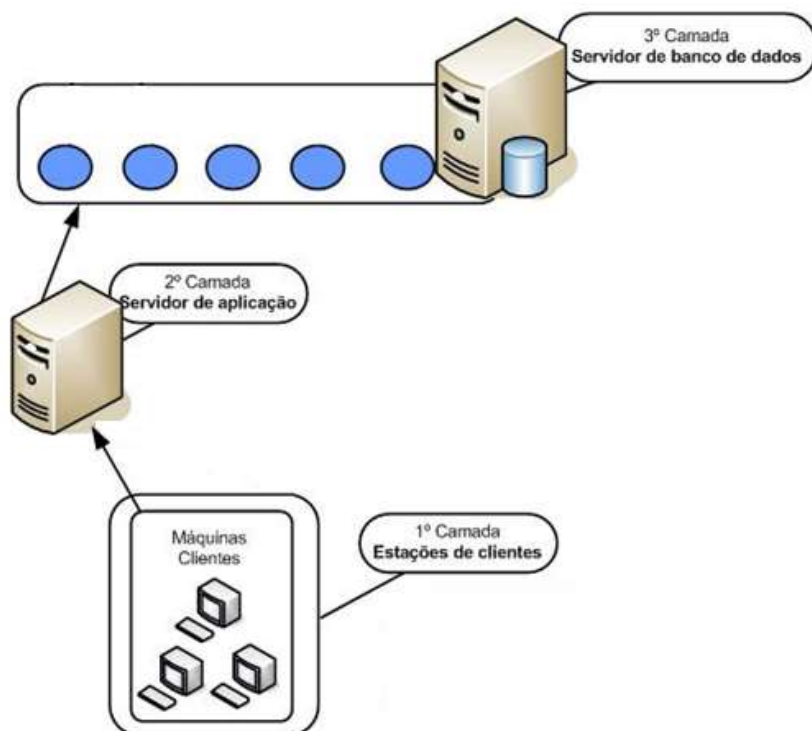
Conectado em dois destes switches temos o Host de Virtualização (SrvHV) que estará com o Sistema Operacional Hypervisor da VMWare, o ESXi 6.5 gerenciando as seguintes VM's:

**SrvWB** – Este será o servidor de Web que armazenará o Apache WEB Server e PHP na última versão estável, utilizando o sistema operacional Linux Ubuntu Server.

**SrvBD** – Este será o servidor de Banco de Dados com sistema operacional Linux Ubuntu Server com banco de dados MySQL na última versão estável.

Como fonte de alimentação ininterrupta teremos 2 Nobreak de 3Kva cada da Apc Smart-Ups que atenderão bem ao nosso cenário.

Para suportar toda esta infraestrutura teremos um Rack de piso fechado nas medidas de 44UX80cmX110cm de cor Preta todo perfurado.



**Figura 1** – Diagrama de Rede

## 7. Implementação

O processo de implementação será da seguinte forma:

Para o sistema de Virtualização, utilizaremos o VMware ESXi.

Para o Servidor WEB, será instalado o sistema operacional Ubuntu Server 16.04, configurado interfaces de rede do servidor, instalar servidor WEB Apache com apt-get, o módulo de segurança, instalar o interpretador PHP e as demais dependências da aplicação (php5 php5-gmp php-pear php5-mysql php5-ldap)

No Servidor de Banco de Dados instalaremos o sistema operacional Ubuntu Server 16.04, configurar interfaces de rede do servidor, adicionar o repositório do MySQL via PPA, instalar pacotes pré-compilados do MySQL com o comando apt-get, configurar senha de root para o banco de dados, criar banco de dados e liberar permissão de acesso ao banco para o servidor WEB

A Aplicação do Wordpress será instalada no servidor frontend e terá o seu código da aplicação clonado no Github no diretório /opt do servidor WEB Apache, será editado a configuração no arquivo config.php da aplicação conforme as informações do banco de dados e o diretório do Virtualhost, depois habilitaremos e configuraremos o módulo rewrite no servidor WEB.

## **8. Solução de virtualização**

Como solução de virtualização utilizaremos o VMware ESXi. Este hypervisor é instalado no servidor físico, e isto permite a divisão em vários servidores lógicos. Este hypervisor possui manutenção fácil devido a sua implementação e configuração serem bem simples de se realizar.

Através do vSphere Command Line Interface (vCLI) e PowerCLI, podemos gerenciar suas tarefas utilizando linhas de comandos remotas usando cmdlets e scripts do Windows PowerShell para gerenciamento automatizado.

Como o ESXi possui um número menor de patches, a sua manutenção é diretamente afetada possuindo assim menos perda de tempo com manutenções desnecessárias.

Para a utilização neste projeto escolhemos o VMware vSphere Essentials Kit, pois este possui 6 licenças de CPU sendo 3 servidores com até dois processadores cada e mais uma licença do VMware vCenter Server Essentials.

Neste kit as licenças não expiram, possui suporte de atualizações e novas versões caso ocorram. Este kit possui um custo de R\$ 1.965,88 já com a versão atual do ESXi 6.5.

## **9. Outras soluções de virtualização**

Fazendo uma avaliação geral com outros sistemas de virtualização mais utilizados no mercado, percebemos que o Hyper-V da Microsoft se aproxima mais do vSphere da VMware, concernente as aplicabilidades de gerenciamento. Porém, a Microsoft fornece gestão de utilidades em inúmeras ferramentas, ao mesmo tempo em que os demais centralizam tudo em um único servidor de gerenciamento.

O XenServer da Citrix, possui um ótimo desempenho trabalhando em conjunto com o Linux, pois contempla uma implantação rápida, porém, umas funções mais avançadas demandam algumas configurações suplementárias. Uma grande desvantagem do XenServer é a sua limitação na gerência e escalabilidade devido sua gestão de VM's serem serializadas, o que torna algumas ações mais simples, como por exemplo o ato de ligar e desligar as máquinas virtuais, um tanto mais que demoradas.

O Red Hat Enterprise Virtualization também possui instalação rápida, porém apresenta alguns problemas relacionados à gestão de host e problemas com alta disponibilidade. Este hypervisor possui desempenho robusto em sistemas Windows e também em sistemas Linux, se tornando o sistema de virtualização mais próximo do VMware concernente a sistemas com componentes de ambientes escaláveis.

Contudo, o sistema de virtualização que melhor atende ao nosso cenário é o VMware vSphere, ainda que a distância entre ele e os seus maiores concorrentes de mercado esteja cada vez menor.



## 10. Referência de downloads

Para a solução de virtualização utilizaremos o VMware ESXi 6.5 onde baixaremos utilizando o link a seguir: <https://my.vmware.com/web/vmware/details?downloadGroup=ESXI650&productId=614>. Como SO das VM's utilizaremos o sistema operacional Ubuntu Server 16.04.2, baixado diretamente do site oficial: <https://www.ubuntu.com/download/server>.

As máquinas virtuais utilizarão como banco de dados o MySQL 5.7.18 adquirido no link: <https://dev.mysql.com/downloads/mysql/>. E um Apache Web Server 2.5 adquirido em <https://httpd.apache.org/docs/trunk/pt-br/>.

Como solução de aplicação, o WordPress, será baixado do site oficial na versão mais atual, que no momento é a 4.7.5 <https://br.wordpress.org/releases/>.

## 11. Infraestrutura de virtualização

Para o sistema de Virtualização, utilizaremos o VMware ESXi 6.5.

Requisitos mínimos para instalação do ESXi:

- Uma plataforma de servidor suportada conforme definido no Guia de Compatibilidade VMware.
- Dois núcleos de CPU.
- Bit NX/XD habilitado para a CPU.
- 4GB de RAM física.
- Recurso de virtualização de hardware Intel VT-x ou AMD RVI para suportar VMs de 64 bits.
- Um ou mais controladores gigabit ou fast ethernet.
- Um disco SCSI local, ou uma LUN RAID diretamente conectada com espaço não-particionado.

Instalação ESXi:





Instalação da aplicação de forma automatizada:

## 12. Planejamento de instalação

1. Primeiro devemos criar um inventário com todas as máquinas envolvidas na instalação, ou seja, com os hosts que irão receber a automatização realizada pelo Ansible, conforme exemplo abaixo:

```
nesello@ubuntu:~$ sudo mkdir wordpress-ansible
nesello@ubuntu:~$ cd wordpress-ansible/
nesello@ubuntu:~/wordpress-ansible$ sudo nano hosts
[sudo] password for nesello:
```

```
nesello@ubuntu:~/wordpress-ansible$ cat hosts
[wpnesello]
192.168.245.135

[dbnesello]
192.168.245.136
nesello@ubuntu:~/wordpress-ansible$
```

2. Agora que possuímos nosso inventário de hosts, vamos criar nosso Playbook Ansible com o seguinte comando:

```
nesello@ubuntu:~/wordpress-ansible$ sudo nano playbook.yml
```

```
nesello@ubuntu:~/wordpress-ansible$ cat playbook.yml
---
- name: Automatizando Wordpress
  hosts: wpnesello
  remote_user: root
  roles:
    - wpnesello

- name: Automatizando DB MySQL
  hosts: dbnesello
  remote_user: root
  roles:
    - dbnesello
nesello@ubuntu:~/wordpress-ansible$
```

3. Neste próximo passo criaremos as “roles” que funciona como plug-ins reutilizáveis. Para a automatização da instalação do WordPress, criaremos 2 roles: A “wpnesello” que armazenará as configurações do nosso servidor e a “dbnesello” que armazenará configurações do banco de dados.

Depois de criarmos o diretório “roles”, entraremos no mesmo e criaremos através de um comando Ansible, uma estrutura com vários arquivos para cada módulo. O arquivo dentro de cada módulo a ser configurado, será basicamente o “tasks/main.yml”.

```
nesello@ubuntu:~/wordpress-ansible$ mkdir roles
nesello@ubuntu:~/wordpress-ansible$ cd roles
nesello@ubuntu:~/wordpress-ansible/roles$ ansible-galaxy init wpnesello
nesello@ubuntu:~/wordpress-ansible/roles$ ansible-galaxy init dbnesello
```

```
nesello@ubuntu:~/wordpress-ansible/roles$ ansible-galaxy init wpnesello
- wpnesello was created successfully
nesello@ubuntu:~/wordpress-ansible/roles$ ansible-galaxy init dbnesello
- dbnesello was created successfully
nesello@ubuntu:~/wordpress-ansible/roles$
```

Após termos a estrutura de arquivos pronta dentro do diretório “roles” vamos editar os arquivos de tarefas de cada role necessária para que a automatização do Ansible funcione adequadamente:

## ROLES WP:

roles/wpnesello/tasks/main.yml

```
---
- name: Atualizando a lista de repositórios APT em WPNESELLO
  apt: update_cache=yes cache_valid_time=3600

- name: Instalando dependências da aplicação WP
  apt: name={{ item }} state=present
  with_items:
    - apache2
    - php7.0-mysql
    - php7.0
    - libapache2-mod-php7.0
    - php7.0-mcrypt
    - python-mysqldb

- name: Instalando extensões PHP
  apt: name={{ item }} state=present
  with_items:
    - php7.0-gd
    - php-ssh2

- name: Baixando o Wordpress
  get_url: url=https://wordpress.org/latest.tar.gz dest=/tmp/wordpress.tar.gz
  validate_certs=no

- name: Descompactando arquivo do Wordpress
  unarchive: src=/tmp/wordpress.tar.gz dest=/var/www/html/ copy=no

- name: Alterando diretório de site padrão do Apache2
  lineinfile: dest=/etc/apache2/sites-enabled/000-default.conf regexp="(.)
+DocumentRoot /var/www/html" line="DocumentRoot /var/www/html/wordpress"
  notify:
    - Reiniciando Servidor Apache2

- name: Copiando modelo de configuração do Wordpress
  command: cp -frv /var/www/html/wordpress/wp-config-sample.php /var/www/html/
wordpress/wp-config.php creates=/var/www/html/wordpress/wp-config.php

- name: Configurando o Wordpress em wp-config.php
  lineinfile: dest=/var/www/html/wordpress/wp-config.php
  regexp="{{ item.regex }}" 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 }}');" }
```

YAML ▾ Tab Width: 8 ▾ Ln 26, Col 1 ▾ INS

roles/wpnesello/handlers/main.yml

```
---
- name: Reiniciando Servidor Apache2
  service: name=apache2 state=restarted
```

roles/wpnesello/defaults/main.yml

```
---
wp_mysql_db: wordpress
wp_mysql_user: wpnesello
wp_mysql_password: SenhaDoDBWP
```

## ROLES DB:

roles/dbnesello/tasks/main.yml

```
---
- name: Atualizando a lista de repositórios APT em DBNESELLO
  apt: update_cache=yes cache_valid_time=3600

- name: Instalando dependências do Banco de Dados MySQL
  apt: name={{ item }} state=present
  with_items:
    - mysql-server

- name: Criando banco de dados do MYSQL
  mysql_db: name={{ wp_mysql_db }} state=present

- name: Adicionando usuário mysql
  mysql_user:
    name={{ wp_mysql_user }}
    password={{ wp_mysql_password }}
    priv=*,*:ALL
```

roles/dbnesello/defaults/main.yml

```
---
wp_mysql_db: wordpress
wp_mysql_user: wpnesello
wp_mysql_password: SenhaDoDBWP
```

Feito isto podemos rodar o comando ansible e aguardarmos o resultado para testarmos através do navegador se a automatização da instalação foi concluída com êxito.

nesello@ubuntu:~/wordpress-ansible\$ ansible-playbook -i ./hosts ./playbook.yml



```

nesello@ubuntu:~/wordpress-ansible$ ansible-playbook -i ./hosts ./playbook.yml

PLAY [Automatizando Wordpress] *****

TASK [setup] *****
ok: [192.168.245.135]

TASK [wpnesello : Atualizando a lista de repositórios APT em WPNESELLO] *****
ok: [192.168.245.135]

TASK [wpnesello : Instalando dependências da aplicação WP] *****
ok: [192.168.245.135] => (item=[u'apache2', u'php7.0-mysql', u'php7.0', u'libapache2-mod-php7.0', u'php7.0-mcrypt', u'python-mysqldb'])

TASK [wpnesello : Instalando extensões PHP] *****
ok: [192.168.245.135] => (item=[u'php7.0-gd', u'php-ssh2'])

TASK [wpnesello : Baixando o Wordpress] *****
ok: [192.168.245.135]

TASK [wpnesello : Descompactando arquivo do Wordpress] *****
ok: [192.168.245.135]

TASK [wpnesello : Alterando diretório de site padrão do Apache2] *****
changed: [192.168.245.135]

TASK [wpnesello : Copiando modelo de configuração do Wordpress] *****
ok: [192.168.245.135]

TASK [wpnesello : Configurando o WordPress em wp-config.php] *****
changed: [192.168.245.135] => (item={u'regexp': u"define\\('DB_NAME', '(.)+'\\);", u'line': u"define('DB_NAME', 'wordpress');"})
changed: [192.168.245.135] => (item={u'regexp': u"define\\('DB_USER', '(.)+'\\);", u'line': u"define('DB_USER', 'wpnesello');"})
changed: [192.168.245.135] => (item={u'regexp': u"define\\('DB_PASSWORD', '(.)+'\\);", u'line': u"define('DB_PASSWORD', 'SenhaDoDBWP');"})

RUNNING HANDLER [wpnesello : Reiniciando Servidor Apache2] *****
changed: [192.168.245.135]

PLAY [Automatizando DB MySQL] *****

TASK [setup] *****
ok: [192.168.245.136]

TASK [dbnesello : Atualizando a lista de repositórios APT em DBNESELLO] *****
ok: [192.168.245.136]

TASK [dbnesello : Instalando dependências do Banco de Dados MySQL] *****
changed: [192.168.245.136] => (item=[u'mysql-server'])

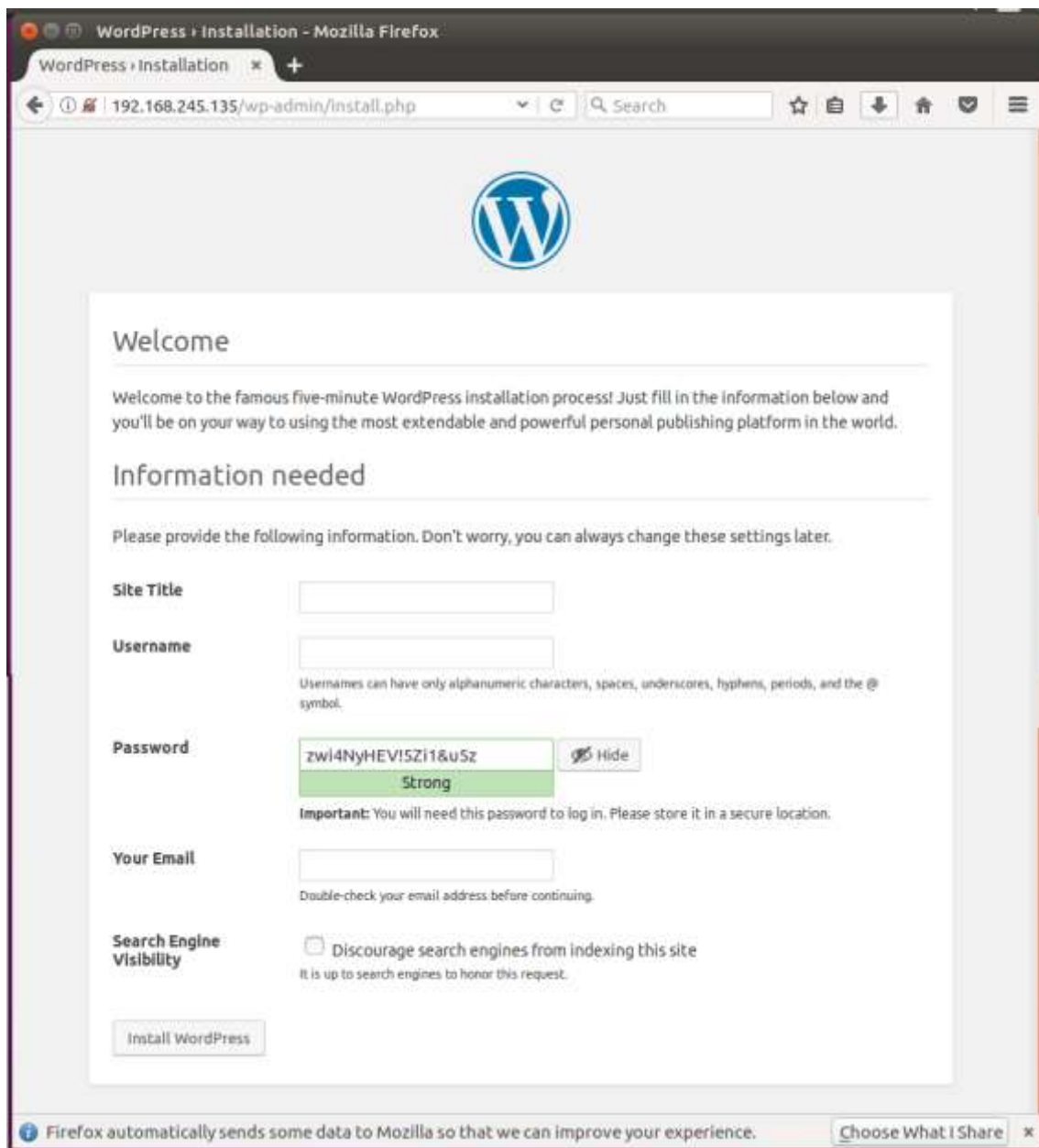
TASK [dbnesello : Criando banco de dados do MYSQL] *****
changed: [192.168.245.136]

TASK [dbnesello : Adicionando usuário mysql] *****
changed: [192.168.245.136]

PLAY RECAP *****
192.168.245.135      : ok=10    changed=3    unreachable=0    failed=0
192.168.245.136      : ok=5     changed=3    unreachable=0    failed=0

nesello@ubuntu:~/wordpress-ansible$

```

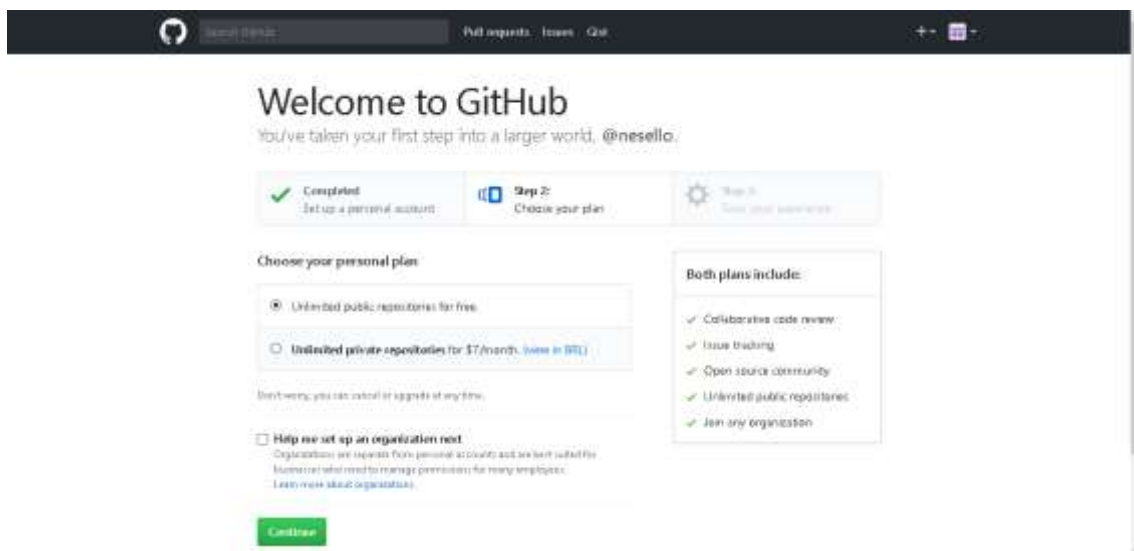
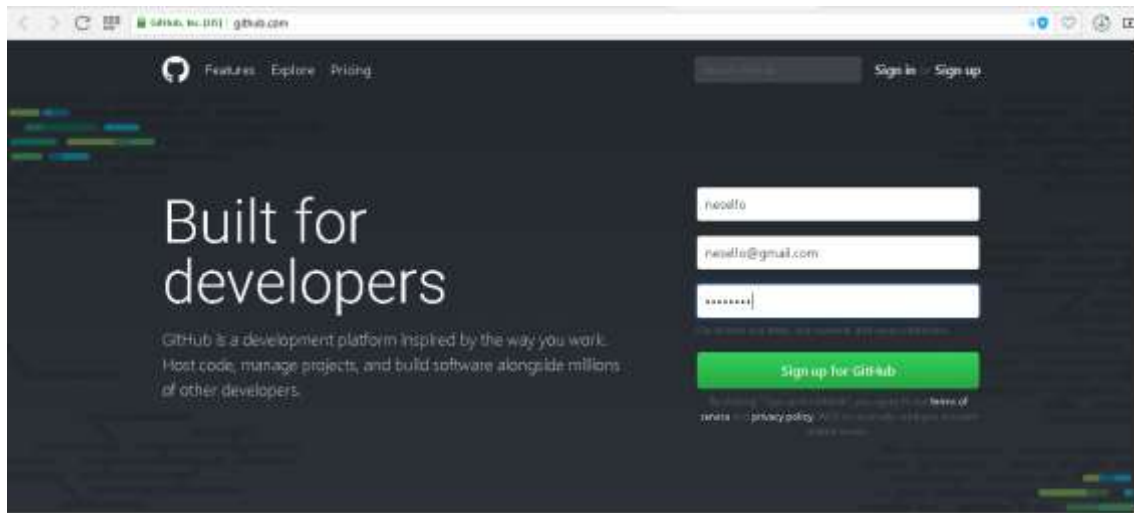


### 13. Documentando repositório

A fim de seguir as boas práticas de mercado vamos elaborar uma documentação e mantê-la atualizada automaticamente, de forma a agilizar a gestão de configuração da própria infraestrutura. Para que isto seja possível iremos utilizar uma ferramenta muito útil na manutenção dos arquivos, scripts e documentação, o Git. Através do Git podemos armazenar arquivos e documentos de uma aplicação, realizar o controle da versão destes documentos e compartilhar com outras pessoas o acesso à estas informações, de forma que os mesmos trabalhem contribuindo em versões diferentes da oficial, cabendo ao administrador adicionar as mudanças ao arquivo original ou não.

## 14. Criando o repositório

Através do gerenciador de pacotes apt(apt-get install git git-core), instalaremos a ferramenta do Git. Feito isso, criaremos uma conta no site github.com para que seja utilizada na criação do repositório. Considere que a chave de autenticação ssh exportada foi add ao github na conta do usuário.





# Welcome to GitHub

You'll find endless opportunities to learn, code, and create, @nesello.

 <b>Completed</b> Set up a personal account	 <b>Step 2:</b> Choose your plan	 <b>Step 3:</b> Tailor your experience
---------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------

How would you describe your level of programming experience?

- ☒ Totally new to programming    ☐ Somewhat experienced    ☐ Very experienced

What do you plan to use GitHub for? (check all that apply)

- ☐ Development    ☐ Project Management    ☐ Design  
☒ School projects    ☐ Research    ☐ Other (please specify)



Which is closest to how you would describe yourself?

- ☒ I'm a student    ☐ I'm a professional    ☐ I'm a hobbyist  
☐ Other (please specify)

What are you interested in?

## Create a new repository



A repository contains all the files for your project, including the revision history.

Owner	Repository name
 nesello ▾	/ wordpress-ansible 

Great repository names are short and memorable. Need inspiration? How about [bug-free-couscous](#).

Description (optional)

Repositório para estudos INFNET

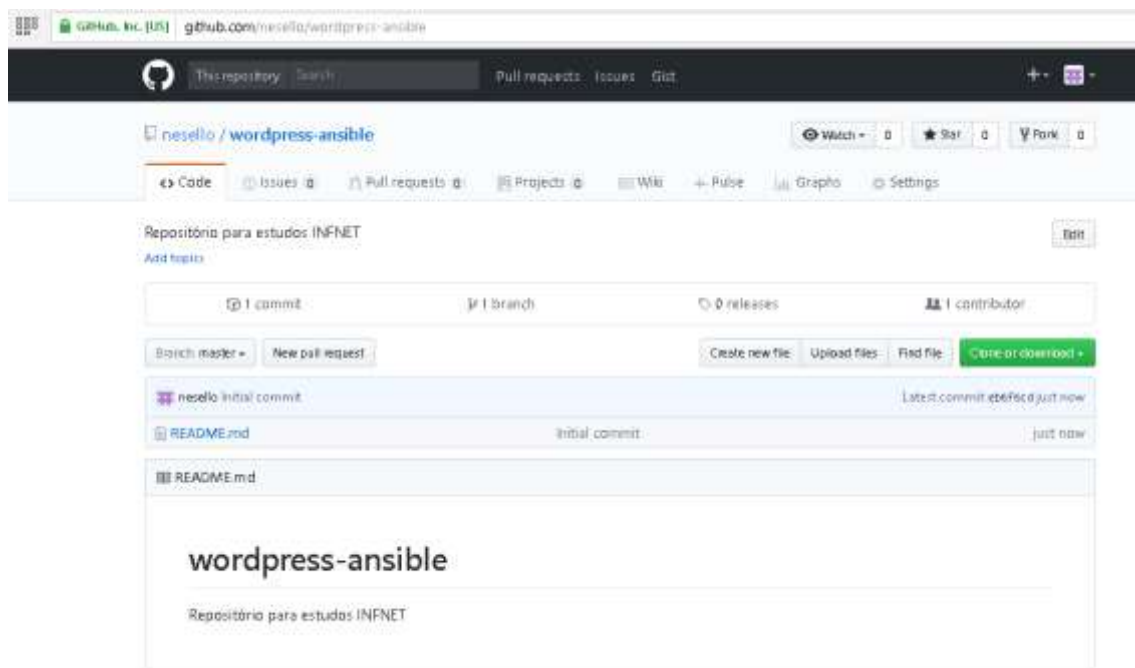
- ☒  **Public**  
Anyone can see this repository. You choose who can commit.
- ☐  **Private**  
You choose who can see and commit to this repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: <b>None</b> ▾	Add a license: <b>None</b> ▾ 
-------------------------------	------------------------------------------------------------------------------------------------------------------

Create repository



Após criarmos a conta, vamos aos comandos para termos acesso ao repositório e adicionarmos os arquivos do playbook ao mesmo:

```
nesello@ubuntu:~$ cd wordpress-ansible
nesello@ubuntu:~/wordpress-ansible$ ls
hosts  playbook.yml  roles
nesello@ubuntu:~/wordpress-ansible$ git init
Initialized empty Git repository in /home/nesello/wordpress-ansible/.git/
nesello@ubuntu:~/wordpress-ansible$ git config --global user.email "nesello@gmail.com"
nesello@ubuntu:~/wordpress-ansible$ git config --global user.name "Rodrigo Nesello"
nesello@ubuntu:~/wordpress-ansible$ git add * # Indexando todos os arquivos do diretório
```

Depois de termos adicionado os arquivos, precisamos realizar o “comit” para confirmar esta adição de arquivos.

```
nesello@ubuntu:~/wordpress-ansible$ git commit -m "Primeiro Commit do Repositorio de Estudos do Ansible"
[master (root-commit) c85d64] Primeiro Commit do Repositorio de Estudos do Ansible
21 files changed, 684 insertions(+)
create mode 100644 hosts
create mode 100644 playbook.retry
create mode 100644 playbook.yml
create mode 100644 roles/dbnesello/.travis.yml
create mode 100644 roles/dbnesello/README.md
create mode 100644 roles/dbnesello/defaults/main.yml
create mode 100644 roles/dbnesello/handlers/main.yml
create mode 100644 roles/dbnesello/meta/main.yml
create mode 100644 roles/dbnesello/tasks/main.yml
create mode 100644 roles/dbnesello/tests/inventory
create mode 100644 roles/dbnesello/tests/test.yml
create mode 100644 roles/dbnesello/vars/main.yml
create mode 100644 roles/wpnesello/.travis.yml
create mode 100644 roles/wpnesello/README.md
create mode 100644 roles/wpnesello/defaults/main.yml
create mode 100644 roles/wpnesello/handlers/main.yml
create mode 100644 roles/wpnesello/meta/main.yml
create mode 100644 roles/wpnesello/tasks/main.yml
create mode 100644 roles/wpnesello/tests/inventory
create mode 100644 roles/wpnesello/tests/test.yml
create mode 100644 roles/wpnesello/vars/main.yml
```

```

nesello@ubuntu:~/wordpress-ansible$ cat .git/config
[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallrefupdates = true
[user]
    email = nesello@gmail.com
    name = Rodrigo Nesello
[remote "origin"]
    url = git@github.com:nesello/wordpress-ansible.git
    fetch = +refs/heads/*:refs/remotes/origin/*

```

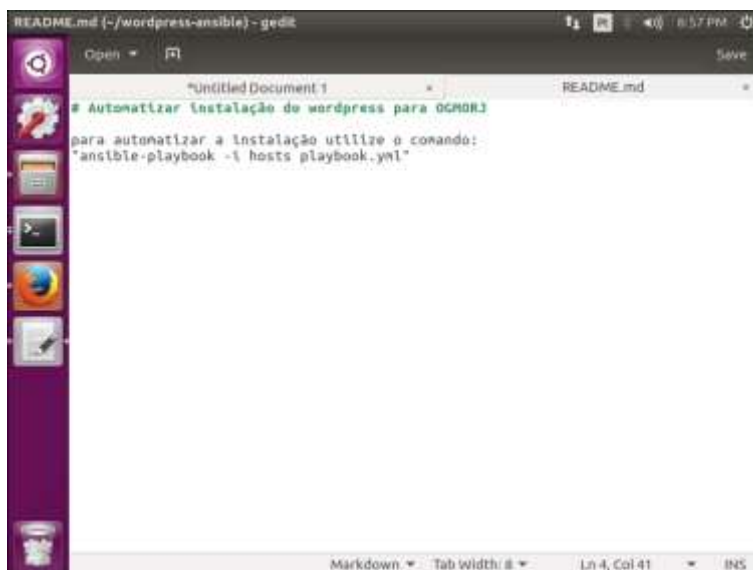
Agora que possuímos um repositório com nossos arquivos de configuração do playbook de atualização do wordpress, vamos atualizar o mesmo a fim de testarmos a atualização do repositório.

```

nesello@ubuntu:~/wordpress-ansible$ git pull origin master
warning: no common commits
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:nesello/wordpress-ansible
 * branch            master       -> FETCH_HEAD
 * [new branch]      master       -> origin/master
Merge made by the 'recursive' strategy.
 README.md | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 README.md

nesello@ubuntu:~/wordpress-ansible$ ls
hosts  playbook.retry  playbook.yml  *README.md*  roles
nesello@ubuntu:~/wordpress-ansible$ git push -u origin master
Counting objects: 33, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (20/20), done.
Writing objects: 100% (33/33), 5.27 KiB | 0 bytes/s, done.
Total 33 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To git@github.com:nesello/wordpress-ansible.git
 6b34814..3793860 master -> master
Branch master set up to track remote branch master from origin.

```



```

nesello@ubuntu:~/wordpress-ansible$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

nesello@ubuntu:~/wordpress-ansible$ git add README.md
nesello@ubuntu:~/wordpress-ansible$ git commit -m "novas instruções no arquivo README"
[master ac54325] novas instruções no arquivo README
1 file changed, 4 insertions(+), 2 deletions(-)

nesello@ubuntu:~/wordpress-ansible$ git push -u origin master
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 417 bytes | 8 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To git@github.com:nesello/wordpress-ansible.git
 3793860..ac54325  master -> master
Branch master set up to track remote branch master from origin.
nesello@ubuntu:~/wordpress-ansible$

```

Após as alterações terem sido realizadas com sucesso, podemos confirmar acessando o repositório através do link <https://github.com/nesello/wordpress-ansible.git> no site github.com com o link de acesso público.

The screenshot shows the GitHub interface for the repository `nesello/wordpress-ansible.git`. At the top, it indicates 4 commits, 1 branch, 0 releases, and 1 contributor. Below this, there are buttons for 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The commit history table shows the following files and their commit times:

File	Commit Message	Time
roles	Primeiro Commit do Repositorio de Estudos do Ansible	an hour ago
README.md	novas instruções no arquivo README	2 minutes ago
hosts	Primeiro Commit do Repositorio de Estudos do Ansible	an hour ago
playbook.retry	Primeiro Commit do Repositorio de Estudos do Ansible	an hour ago
playbook.yml	Primeiro Commit do Repositorio de Estudos do Ansible	an hour ago

The README.md content is displayed below the table, with the title **Automatizar instalação do wordpress para OGMORJ** and the instruction: `para automatizar a instalação utilize o comando: "ansible-playbook -i hosts playbook.yml"`.

## 15. Aplicação distribuída com dois containers Docker

Como aplicação distribuída iremos utilizar o Wordpress, pois é a aplicação que estamos utilizando desde o início do projeto. Porém, faremos sua instalação através de containers Docker.

Para isto, precisamos instalar antes o Docker na nossa máquina virtual (VM está somente com o Ubuntu instalado) conforme os passos a seguir:



```

nesello@ubuntu:~$ sudo apt-get install apt-transport-https ca-certificates curl
software-properties-common
Reading package lists... Done
Building dependency tree
Reading state information... Done
ca-certificates is already the newest version (20160104ubuntu1).
apt-transport-https is already the newest version (1.2.19).
software-properties-common is already the newest version (0.96.20.5).
The following NEW packages will be installed:
  curl
0 upgraded, 1 newly installed, 0 to remove and 3 not upgraded.
Need to get 139 kB of archives.
After this operation, 338 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 curl amd64 7
.47.0-1ubuntu2.2 [139 kB]
Fetched 139 kB in 2s (50.0 kB/s)
Selecting previously unselected package curl.
(Reading database ... 174305 files and directories currently installed.)
Preparing to unpack .../curl_7.47.0-1ubuntu2.2_amd64.deb ...
Unpacking curl (7.47.0-1ubuntu2.2) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up curl (7.47.0-1ubuntu2.2) ...
nesello@ubuntu:~$

```

Com este primeiro comando, iremos instalar pacotes que irão executar funções de assinatura digital e também de download dos pacotes necessários para o funcionamento de toda estrutura do Docker.

```

nesello@ubuntu:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
  apt-key add -
OK
nesello@ubuntu:~$

```

O comando demonstrado acima executa o download da chave de assinatura digital do projeto Docker, permitindo a autenticação dos programas a serem baixados do mesmo. Também estamos neste mesmo comando adicionando o gerenciador de pacotes.

```

nesello@ubuntu:~$ sudo apt-key fingerprint 0EBFCD88
pub  4096R/0EBFCD88 2017-02-22
    Key fingerprint = 9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
uid          Docker Release (CE deb) <docker@docker.com>
sub  4096R/F273FCD8 2017-02-22
nesello@ubuntu:~$

```

Através deste comando iremos expor a identificação da chave que foi adicionada com o comando anterior a este em questão.

```

nesello@ubuntu:~$ sudo add-apt-repository "deb [arch=amd64] https://download.doc
ker.com/linux/ubuntu $(lsb_release -cs) stable"
nesello@ubuntu:~$

```

Agora estaremos adicionando o repositório de pacotes do projeto Docker. Com isto será possível baixar e instalar os softwares que estão disponíveis no projeto, inclusive o Docker em questão.

```

nesello@ubuntu:~$ sudo apt-get update
Hit:1 http://us.archive.ubuntu.com/ubuntu xenial InRelease
Get:2 https://download.docker.com/linux/ubuntu xenial InRelease [20.1 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
Get:4 http://security.ubuntu.com/ubuntu xenial-security InRelease [102 kB]
Get:5 https://download.docker.com/linux/ubuntu xenial/stable amd64 Packages [1,479 B]
Get:6 http://us.archive.ubuntu.com/ubuntu xenial-backports InRelease [102 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 DEP-11 Metadata [288 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu xenial-updates/main DEP-11 64x64 Icons [190 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu xenial-updates/universe amd64 DEP-11 Metadata [160 kB]
Get:10 http://security.ubuntu.com/ubuntu xenial-security/main amd64 DEP-11 Metadata [54.2 kB]
Get:11 http://us.archive.ubuntu.com/ubuntu xenial-updates/universe DEP-11 64x64 Icons [186 kB]
Get:12 http://security.ubuntu.com/ubuntu xenial-security/main DEP-11 64x64 Icons [42.4 kB]
Get:13 http://security.ubuntu.com/ubuntu xenial-security/universe amd64 DEP-11 Metadata [32.2 kB]
Get:14 http://us.archive.ubuntu.com/ubuntu xenial-updates/multiverse amd64 DEP-11 Metadata [2,516 B]
Get:15 http://us.archive.ubuntu.com/ubuntu xenial-backports/main amd64 DEP-11 Metadata [3,328 B]
Get:16 http://security.ubuntu.com/ubuntu xenial-security/universe DEP-11 64x64 Icons [33.0 kB]
Fetched 1,320 kB in 8s (159 kB/s)
Reading package lists... Done
nesello@ubuntu:~$

```

Através do update, estaremos baixando os pacotes disponíveis nos repositórios configurados. Este comando é fundamental para se obter os pacotes cedidos através do repositório Docker, recentemente adicionado.



```

nesello@ubuntu:~$ sudo apt-get install docker-ce
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  aufs-tools cgroupfs-mount git git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-arch git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  aufs-tools cgroupfs-mount docker-ce git git-man liberror-perl
0 upgraded, 6 newly installed, 0 to remove and 3 not upgraded.
Need to get 23.1 MB of archives.
After this operation, 115 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/universe amd64 aufs-tools amd64
  1:3.2+20130722-1.1ubuntu1 [92.9 kB]
Get:2 https://download.docker.com/linux/ubuntu xenial/stable amd64 docker-ce amd
  64 17.03.1~ce-0~ubuntu-xenial [19.3 MB]
Get:3 http://us.archive.ubuntu.com/ubuntu xenial/universe amd64 cgroupfs-mount a
  ll 1.2 [4,970 B]
Get:4 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 liberror-perl all 0.
  17-1.2 [19.6 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 git-man all 1:2.7.4-
  0ubuntu1 [735 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 git amd64 1:2.7.4-0u
  buntu1 [3,006 kB]
Fetched 23.1 MB in 12s (1,859 kB/s)
Selecting previously unselected package aufs-tools.
(Reading database ... 174312 files and directories currently installed.)

```

Com este comando daremos início a instalação do sistema Docker, tornando possível a execução dos containers do MySQL e do Wordpress.

```

nesello@ubuntu:~$ sudo docker run --name banco -e MYSQL_ROOT_PASSWORD=senha123 -
d mysql:5.6
3910cd3c00551639f493e4845a83a94e3905d66a44c52873e4df0c937dd1714c
nesello@ubuntu:~$

```

Agora que temos o Docker instalado em nosso sistema, iremos “rodar” um novo container de banco de dados MySQL, onde este receberá variáveis de configuração para definição de “senha root”. Este Container funcionará como um serviço rodando em segundo plano.

```

nesello@ubuntu:~$ sudo docker ps

```

CONTAINER ID	IMAGE	COMMAND	CREATED
3910cd3c0055	mysql:5.6	"docker-entrypoint..."	About a minute ago
Up About a minute	3306/tcp	banco	

A fim de corroborar se o container está realmente sendo executado, basta utilizarmos o comando acima.

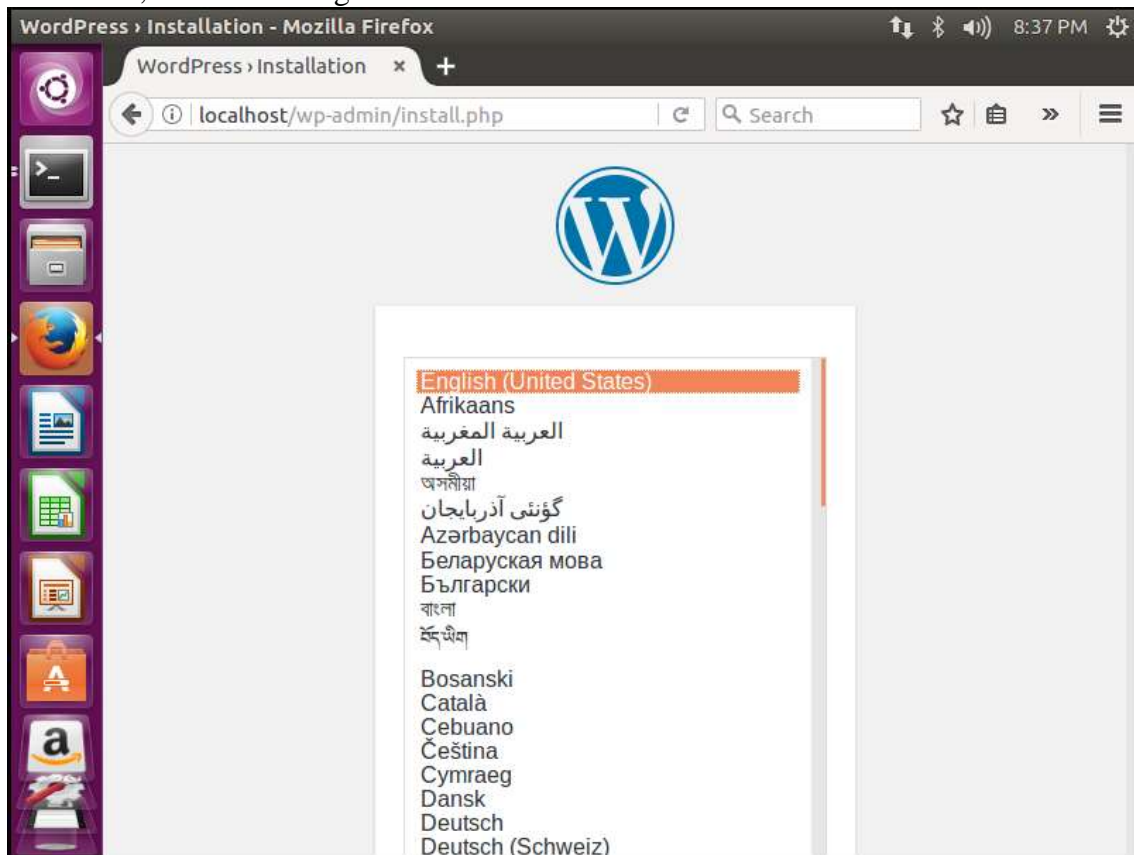
Comprovada a instalação e o funcionamento do container do MySQL, poderemos prosseguir com a instalação do container do WordPress.

```
nesello@ubuntu:~$ sudo docker run --name meusite --link banco:mysql -p 80:80 -d  
wordpress  
818aa929cac08247707d51a3d430937f584ce04ed55128751164e4aca6efb102  
nesello@ubuntu:~$
```

Com o comando acima, estaremos executando o segundo container que no caso é o do Wordpress. Neste container faremos referência ao container do banco de dados MySQL para que o Wordpress possa recuperar a senha de root, podendo se conectar ao mesmo sem a necessidade de configurações extras.

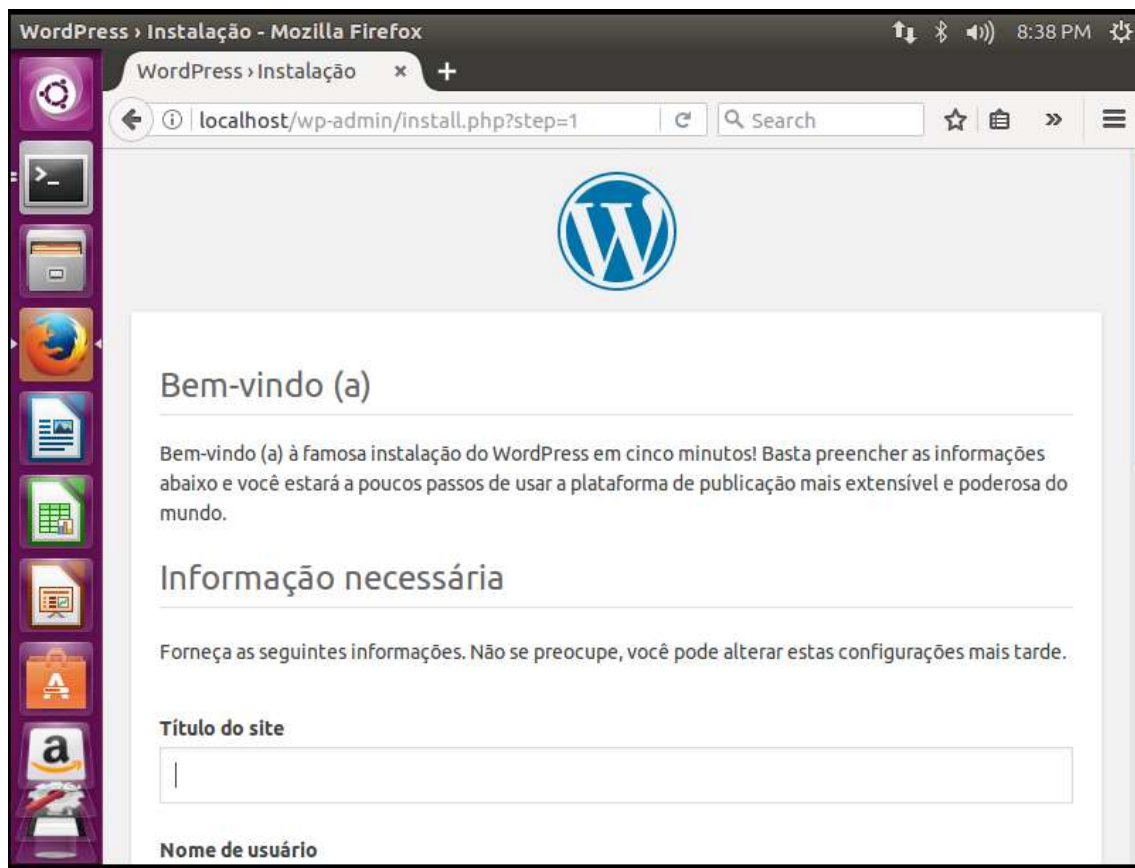
Através do comando “-p 80:80” estamos dizendo que ao ocorrer uma conexão na porta 80 do computador, esta conexão será redirecionada para a porta 80 do container e a parte do comando “-d wordpress” carrega o container em segundo plano baixando e executando o Wordpress.

A fim de testarmos se os containers estão funcionando, iremos através de um computador cliente, acessar o Wordpress que acabou de ser instalado através de containers, conforme imagens abaixo:



Acesso local





Acesso local

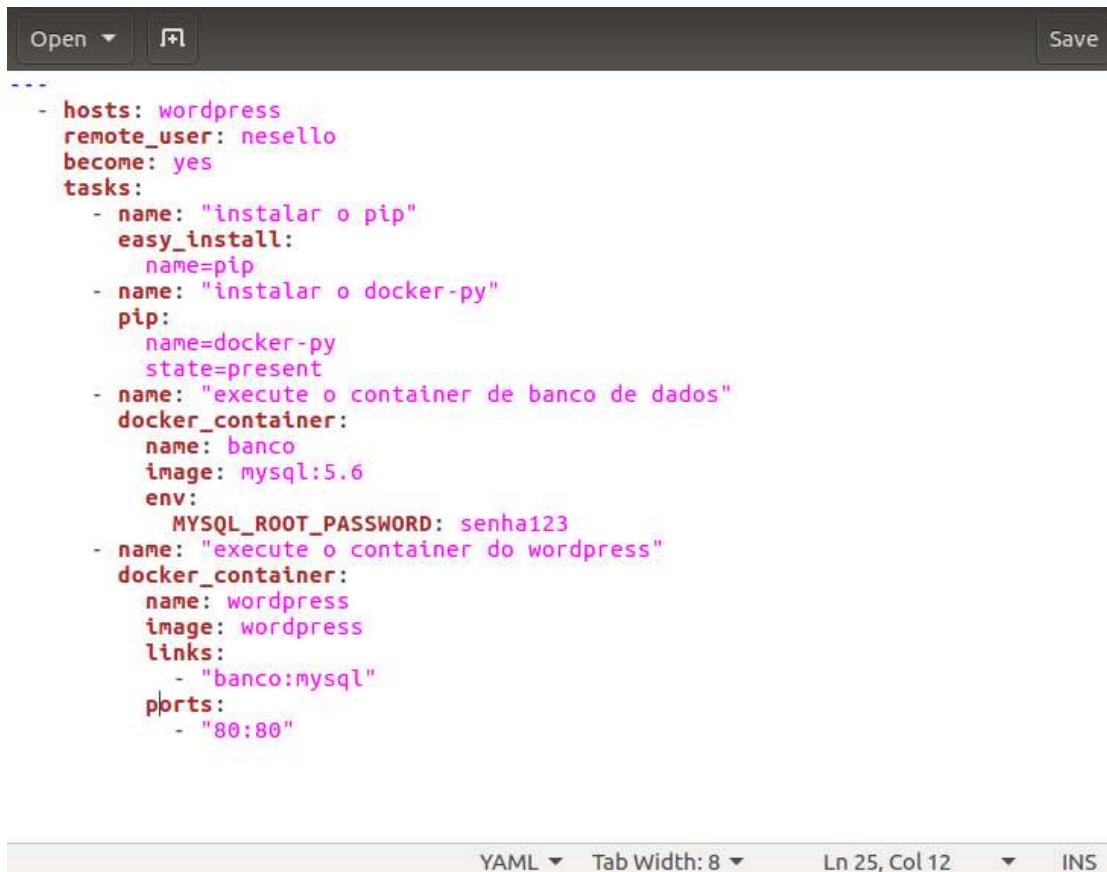


Acesso a partir de cliente

## 16. Automatizando instalação de Containers

Agora veremos como realizar toda configuração vista no item 15, porem de forma automatizada através de um Playbook Ansible com dois containers. Como aplicação distribuída continuaremos utilizando o Wordpress, conforme estamos utilizando desde o início do projeto.

Portanto, vamos ao Playbook:



```
---
- hosts: wordpress
  remote_user: nesello
  become: yes
  tasks:
    - name: "instalar o pip"
      easy_install:
        name=pip
    - name: "instalar o docker-py"
      pip:
        name=docker-py
        state=present
    - name: "execute o container de banco de dados"
      docker_container:
        name: banco
        image: mysql:5.6
        env:
          MYSQL_ROOT_PASSWORD: senha123
    - name: "execute o container do wordpress"
      docker_container:
        name: wordpress
        image: wordpress
        links:
          - "banco:mysql"
        ports:
          - "80:80"
```

YAML ▾ Tab Width: 8 ▾ Ln 25, Col 12 ▾ INS

Primeiro devemos criar o Playbook com os comandos necessários para a instalação do container do SQL e do container do Wordpress, conforme imagem acima.

```
nesello@ubuntu: ~/wordpress-ansible
nesello@ubuntu:~/wordpress-ansible$ ansible-playbook -i hosts playbook.yml

PLAY [wordpress] *****

TASK [setup] *****
ok: [127.0.0.1]

TASK [instalar o pip] *****
ok: [127.0.0.1]

TASK [instalar o docker-py] *****
ok: [127.0.0.1]

TASK [execute o container de banco de dados] *****
ok: [127.0.0.1]

TASK [execute o container do wordpress] *****
ok: [127.0.0.1]

PLAY RECAP *****
127.0.0.1 : ok=5 changed=0 unreachable=0 failed=0

nesello@ubuntu:~/wordpress-ansible$
```

Após a preparação do Playbook, entramos com o comando `ansible-playbook -i hosts playbook.yml` para executarmos a instalação dos containers, conforme mostrado na imagem acima.



Depois de termos rodado o ansible com seus devidos containers, podemos realizar um teste de acesso através de um PC cliente, conforme imagem acima.

```
nesello@ubuntu: ~/wordpress-ansible
nesello@ubuntu:~$
nesello@ubuntu:~$ cd wordpress-ansible/
nesello@ubuntu:~/wordpress-ansible$ ls
hosts  playbook.retry  playbook.yml  roles
nesello@ubuntu:~/wordpress-ansible$ git init
Reinitialized existing Git repository in /home/nesello/wordpress-ansible/.git/
nesello@ubuntu:~/wordpress-ansible$ git add *
nesello@ubuntu:~/wordpress-ansible$ git commit -m "TP5"
On branch master
nothing to commit, working directory clean
nesello@ubuntu:~/wordpress-ansible$ git remote add origin https://github.com/nesello/TP5
nesello@ubuntu:~/wordpress-ansible$ git push -u origin master
Username for 'https://github.com': nesello
Password for 'https://nesello@github.com':
Counting objects: 55, done.
Compressing objects: 100% (19/19), done.
Writing objects: 100% (55/55), 5.81 KiB | 0 bytes/s, done.
Total 55 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/nesello/TP5
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
nesello@ubuntu:~/wordpress-ansible$
```

Já que o projeto do Ansible com a instalação dos containers foi bem-sucedida, iremos “comitar” nosso projeto no repositório do Github para fins de backup e utilizações futuras para novas instalações, através do link: <https://github.com/nesello/TP5>

## 17. Cronograma de instalação

Levando em consideração que toda a infraestrutura de hardware se encontra pronta e disponível, apresento abaixo o cronograma de instalação da solução pretendida.

- Instalação do ESXi 6.5 – entre 15 e 20 minutos.
- Criação de VM’s com SO Linux Ubuntu – entre 20 e 30 minutos.
- Implementação do WordPress via Ansible – entre 30 e 40 minutos.

Todo o processo de implementação levará entre 65 e 90 minutos para a sua conclusão.

## 18. Conclusão

O prazo estabelecido para o projeto foi adequado para a execução do mesmo, bem como os seus recursos que também foram suficientes para atender o projeto como planejado, possibilitando pôr a solução em funcionamento.

Todas as funcionalidades previstas no projeto, funcionaram conforme o esperado permitindo que sua implementação fosse concluída com sucesso.

A fim de melhorar o cenário atual da empresa, eu recomendo para o futuro próximo, a migração de todo o data center para a nuvem da AWS, pois assim poderíamos abrir mão da preocupação com hardware, e focar mais tempo na parte de desenvolvimento. Poderíamos também, contar mais com redundância, balanceamento de carga, e otimização de custos, pagando apenas pelos serviços utilizados e/ou consumidos.

## 19. Referências

<http://www.vmware.com/br.html>

[https://codex.wordpress.org/pt-br:Página\\_Inicial](https://codex.wordpress.org/pt-br:Página_Inicial)

<https://br.wordpress.org>

<https://www.digitalocean.com/community/tutorials/how-to-automate-installing-wordpress-on-ubuntu-14-04-using-ansible>

<https://www.digitalocean.com/community/tutorials/how-to-install-wordpress-on-ubuntu-14-04>

<https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu>

[http://store.vmware.com/store/vmwbr/pt\\_BR/cat/categoryID.67818600](http://store.vmware.com/store/vmwbr/pt_BR/cat/categoryID.67818600)

[http://rogerdudler.github.io/git-guide/index.pt\\_BR.html](http://rogerdudler.github.io/git-guide/index.pt_BR.html)

<https://github.com/nesello/Projeto-Final>