



Instituto Infnet

CURSO DE GRADUAÇÃO EM REDE DE
COMPUTADORES

**Projeto de Bloco: Arquitetura e Infraestrutura de
Aplicações**

ENTREGA DE PROJETO:

Documentando sua infraestrutura de nuvem

Daniel Felipe Meireles de Souza

Turma: RDC2017 – Manhã 1

Professor: Carlos Nilton

Sumário

Informações Gerais.....	4
Introdução.....	5
1. Caso de Estudo: GIRO Telecomunicações LTDA.....	6
1.1. A Aplicação: phpIPAM.....	7
1.2. Infraestrutura.....	9
1.3. SDDC: Software Defined Data Center.....	10
1.4. Estimativa de Recursos.....	11
1.5. Plano de Implantação.....	12
1.6. Definições de Rede.....	14
1.7. Alternativas de Implementação.....	15
1.7.1. Exemplo AWS.....	15
1.7.2. Exemplo OpenStack.....	16
1.7.4. Analisando as Opções.....	19
1.8. Preparando a Solução em Nuvem.....	20
1.8.1. Iniciando a instalação.....	20
1.8.2. Iniciando Servidor ESXi.....	22
1.9. Instalando vCenter.....	24
1.9.1. Resolvendo dependências.....	24
1.9.2. Iniciando a instalação do vCenter.....	26
1.9.3. Criando Data Center Virtual.....	29
2. Utilização de ferramentas para automação de instalação de aplicações.....	31
2.1. Ansible.....	31
2.2. Criando Playbooks.....	32
2.3. Executando as instruções Ansible.....	34
2.4. Aplicação em Funcionamento.....	36
3. Documentar um projeto de infraestrutura para aplicações.....	37
3.1. Criando Repositório Git.....	38
3.2. Realizando o <i>Push</i> das Alterações no Repositório.....	39
4. Selecionando a plataforma de execução para uma aplicação.....	42
5. Gerenciamento de nuvem com Ansible.....	44
5.1. Role docker (Ativando suporte ao docker).....	44

5.2. Role mysql-docker (Rodando o Contêiner MySQL).....	45
5.3. Role phpipam-docker (Rodando o Contêiner phpIPAM).....	45
5.4. Playbook ansible docker-playbook.yml.....	46
5.5. Atualizando a Documentação com o GIT.....	49
Conclusão.....	50
Referências e Bibliografias.....	52

Informações Gerais

O presente documento tem como objetivo demonstrar de maneira prática e objetiva, os conceitos de *Arquitetura e Infraestrutura de Aplicações*. Nele serão apresentadas aplicações conceituais e práticas aprendidas nas disciplinas regulares, *Fundamentos da Arquitetura de Infraestrutura de Aplicações* e *Virtualização de Desktop e Aplicações*, além dos entendimentos absorvidos pela disciplina do projeto de bloco *Arquitetura e Infraestrutura de Aplicações*.

Na primeira etapa do projeto de bloco será proposta uma solução de arquitetura de infraestrutura para uma aplicação de acordo com um cenário conceitual.

Na etapa dois é demonstrado a utilização de ferramentas de automação de instalação de aplicações com o *ansible*. Com base no conceito do *ansible* foram criados modelos de instalação para os servidores propostos para o projeto.

Na terceira etapa são apresentados os conceitos de documentação do projeto de infraestrutura para aplicações. Nesta fase foi utilizado o sistema de armazenamento de código-fonte Git, que é uma ferramenta para edição de documentos de forma colaborativa e distribuída.

Na quarta etapa estão sendo apresentados os conceitos dos contêineres de aplicações e sua instalação no cenário proposto. Esta implementação entrará como uma alternativa para a aplicação distribuída, mas será realizada a partir de contêineres da aplicação docker.

Na quinta parte do projeto é demonstrado um modelo de orquestração de servidores com a utilização da ferramenta *Ansible*, gerenciando os servidores em nuvem com esta aplicação.

Introdução

A interação dos usuários com os diversos sistemas de informação hoje é crucial para o sucesso de qualquer negócio. Agilizar e automatizar processos, que antes eram manuais e pouco produtivos, são atribuições que os sistemas de informação integrados devem fazer, isso fora o fato de que quanto mais automatizado menores são as chances de ocorrerem erros humanos.

Este trabalho objetiva-se a demonstrar conceitos e metodologias modernas dos sistemas de TI atuais, tais como Virtualização ou *cloud computing*. Ao longo deste documento serão apresentados os conceitos de maneira prática e objetiva em um cenário contextualizado.

A automação de processos de instalação de aplicações também foi utilizada dentro do contexto. Com ferramentas deste tipo é possível agilizar a instalação de servidores customizados de acordo com a demanda e em larga escala.

Para a documentação e controle de versões dos códigos da aplicação será utilizado o GIT, que é um *VCS (Version control system)* enfatizado na velocidade. Com o GIT, os *playbooks* escritos para o *ansible* etapa 2 serão documentados e publicados para que possam ser desenvolvidos colaborativamente no futuro.

Diversos cenários podem ser descritos para as aplicações distribuídas, e um deles é a utilização de aplicações pré desenhadas e configuradas, como é o caso de aplicações em contêineres muito utilizadas em computação em nuvem. Ferramentas como essas são muito úteis do ponto de vista da facilidade de implementação e escalonamento, por isso serão apresentadas como alternativa à implementação realizada anteriormente com a automação do *ansible*.

A orquestração dos múltiplos componentes de uma aplicação distribuída em nuvem, pode ser realizada de maneira magistral com ferramentas voltadas para esse fim. O *Ansible* é uma ótima resposta, para o administrador de uma infraestrutura distribuída, na questão do controle em massa dos componentes da nuvem.

1. Caso de Estudo: GIRO Telecomunicações LTDA.

Uma nova empresa de provisão de serviços de internet(ISP) chamada *GIRO Telecomunicações*, recebeu outorga da ANATEL para comercializar Serviços de Comunicação multimídia, o que inclui internet.

Enquanto haviam os trâmites legais para a liberação da outorga, a empresa já havia entrado com um pedido para a entidade responsável pelas atividades de registro e manutenção de nomes de domínio com sufixo “.br” e distribuição de recursos de numeração de sistemas autônomos(ASN) e endereçamento IPv4 e IPv6 do Brasil, também conhecida como REGISTRO.BR.

A entidade liberou para a GIRO um número de sistema autônomo contendo uma rede de endereços IPv6 com um prefixo de 64bits e uma rede de endereços IPv4 com prefixo de 22bits.

A rede IPv6 cedida pelo RegistroBR é grande o bastante para atender a demanda inicial da empresa, todavia o endereçamento IPv4 liberado é bem pequeno em se tratando de um ISP, mas com a escassez dos endereços deste tipo, não havia como receber um prefixo com maior número de endereços, o que se fez necessário um plano de gerenciamento de endereços.

O plano de gerenciamento de endereços consiste em catalogar cada rede ou *host* concedidos para os clientes da GIRO Telecom. Para isso, será arquitetada uma aplicação a ser usada na infraestrutura da empresa, que gerencie e automatize os processos de: solicitação de endereçamento, registro de endereços e clientes a eles associados, registro e gerenciamento de sistemas autônomos e outras funções de gerenciamento de endereço IP. Esta aplicação ajudará a empresa a controlar os endereços de redes utilizados pelos clientes, tanto IPv4 quanto IPv6, agilizando os processos do tipo e prevenindo o desperdício de endereços.

O plano da GIRO é ter esta plataforma de controle de endereços pronta antes do lançamento da marca e seus produtos.

1.1. A Aplicação: phpIPAM

Tendo em vista a necessidade da empresa, foi feito um trabalho de pesquisa para encontrar a melhor ferramenta para cobrir esta demanda de documentar os endereços. A escolha de uma ferramenta Open Source foi a mais indicada para o momento em que a empresa se encontra. Minimizar os custos sempre é uma boa opção quando se está começando, e ferramentas de código aberto são sempre opções boas nesse quesito.



Figura 1. phpIPAM

Categoria: HTML/XHTML, Banco de Dados, AJAX

Descrição:

O phpIPAM(Personal Home Page IP Address Manager) é uma aplicação WEB de código aberto para gerenciamento de endereços IP. Seu objetivo é fornecer um gerenciamento de endereços IP leve, moderno e útil. É uma aplicação baseada em php com backend de banco de dados MySQL, usando bibliotecas jQuery, ajax e alguns recursos HTML5 / CSS3 para a exibição da interface de usuário.

Características: Com front-end desenvolvido inicialmente na linguagem *PHP* e *HTML*, esta aplicação pode ser utilizada com *Postgresql* ou *MySQL* como banco de dados em seu back-end. Possui também alguns *shells scripts* para coleta de informações de ICMP e SNMP sobre os endereços e a disponibilidade de *redes hosts e novas redes*.

Versionamento: A versão atual da aplicação é *1.2.1* e o controle de versões é realizado por *Miha Petkovsek*(miha.petkovsek@gmail.com) principal desenvolvedor da aplicação pela phpIPAM.

Desenvolvimento: Atualmente o desenvolvimento da ferramenta é *Open Source* e colaborativo sendo possível encontrar o código no GitHub além do site oficial e do sourceforge.net. Nesta plataforma, *GitHub*, colaboradores remotos podem verificar o código e até mesmo propor melhorias para os desenvolvedores.

Licenciamento: *GNU General Public License version 3.0 (GPLv3)*

Recursos: Gerenciamento de Endereços IPv4 / IPv6; Atualização de status ICMP; escaneamento e exibição de status de sub-redes, Autenticação em domínio (AD) / OpenLDAP; Permissões por grupo; Multiple level of nested subnets; Exibição visual de subredes; Calculadora de endereçamento IPv4/IPv6; suporte a VRF; Gerenciamento de VLAN; Gerenciamento de dispositivos; importação de tabela RIPE; Importa e exporta arquivos XLS/CSV; Gerenciamento de usuários; Notificação via e-mail com detalhes de IP; busca em banco de dados de IP; Módulo de solicitação de IP; Adição/Remoção/Edição de faixas de endereços IP; Campos customizados de IP/subnet/user/VLAN.

Dependências:

1. Servidor Apache com suporte a PHP;
2. Servidor MySQL(5.1+);
3. PHP versão 5.3 ou superior com os seguintes módulos habilitados: pdo, pdo_mysql, sockets, openssl, gmp, ldap, crypt, SimpleXML, json, gettext, filter, pcntl, cli, mbstring
4. Suporte a php PEAR

1.2. Infraestrutura

Para o funcionamento da aplicação é necessário atender a algumas dependências como visto anteriormente. Pensando nisto a infraestrutura para atender as necessidades da aplicação foram definidas de acordo com o diagrama abaixo e as necessidades da aplicação:

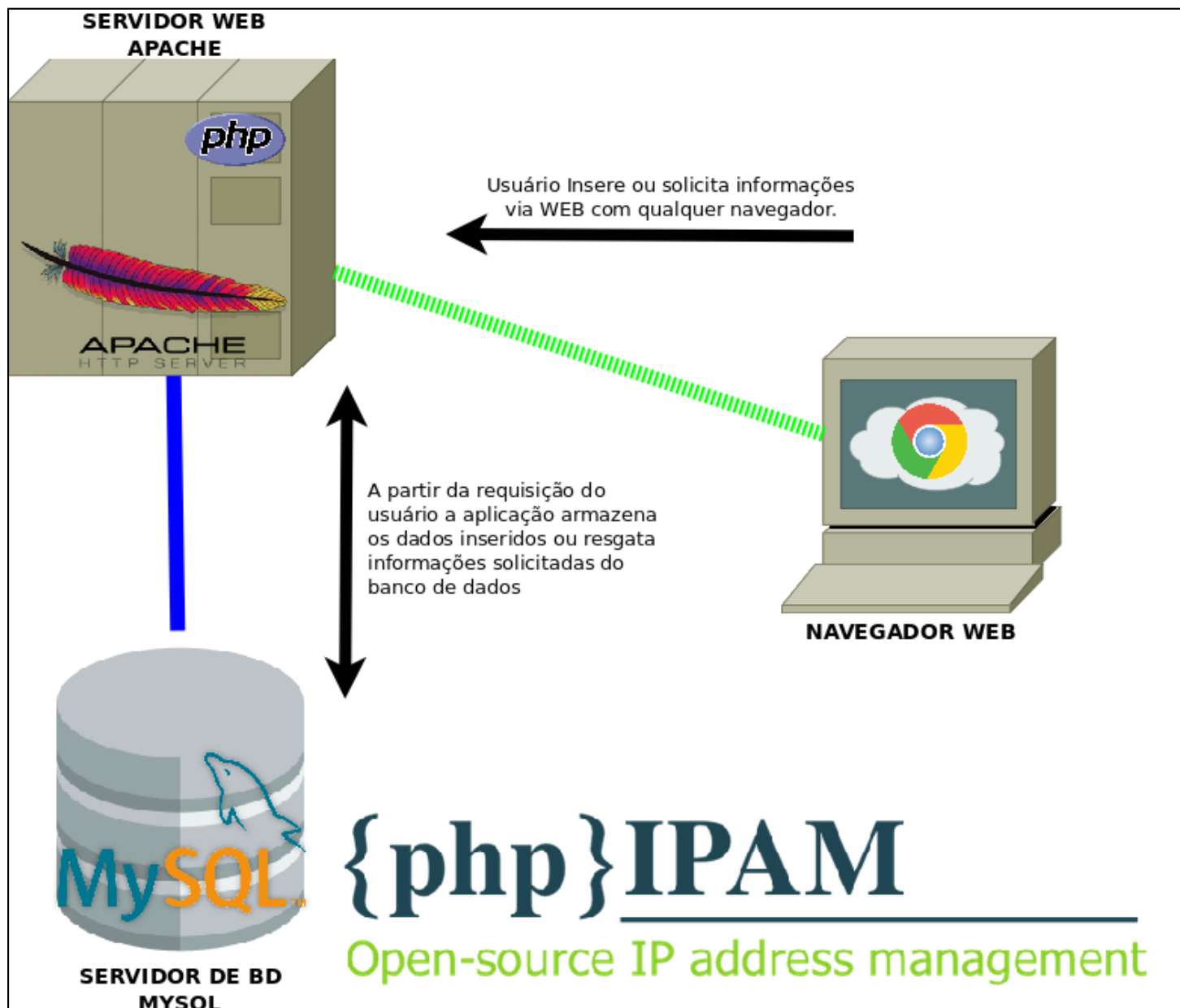


Figura 2. Diagrama da Aplicação

Os usuários interagem com a aplicação utilizando qualquer tipo de navegador WEB. A base da infraestrutura para a aplicação compreende em dois servidores, um para o *backend* banco de dados e outra para o *frontend* da página PHP no servidor apache.

1.3. SDDC: Software Defined Data Center

Essa infraestrutura de dois servidores, mesmo que simples, pode ser problemática para gerenciar cada servidor individualmente. Pensando nisso foi elaborado um plano de implementação utilizando o modelo SDDC(Data Center Definido por Software).

Com este modelo, é possível destacar quatro vantagens na gestão dos serviços agregados a aplicação:

1. Virtualização: Facilitação da Gestão

Com a tecnologia de Data Center definido por software toda a infraestrutura é virtualizada e as operações são fornecidas na forma de serviço, permitindo a sustentabilidade e agilidade dos negócios. Este modelo aplica os conceitos de abstração para entregar uma estrutura de Data Center inteira em software, separando o fornecimento de serviços da infraestrutura.

2. Automação e Flexibilidade

A possibilidade de aumentar o nível de automação do ambiente flexibilizando às escolhas e ao atendimento da demanda dos negócios é outra grande vantagem do modelo. Baseando-se em políticas definidas de automação de rotinas, como o provisionamento, a localização, configuração e controle, reduz substancialmente parte da intervenção humana com esta tecnologia. Assim os serviços de Data Center tornam-se bem menos complexos.

3. Plataforma de Nuvem Híbrida

Atualmente os fornecedores trabalharam criando ofertas de Nuvens Públicas independentes sendo desenvolvidas com o uso da tecnologia SDDC baseada em produtos de virtualização. Integrando-se uma Nuvem Privada para uma nuvem pública desenvolvida com o modelo SDDC, a configuração do Data Center torna-se uma verdadeira plataforma de Nuvem Híbrida abrangendo ambientes dentro ou fora dele. Dessa forma, uma das maiores vantagens de Data Center definido por Software é que não se precisa de um hardware especializado.

4. Segurança

No modelo SDDC também foi descoberto, de forma não intencional pelas empresas, é a questão da segurança. Devido a abordagem que pode ser encarada como uma ferramenta pelas equipes de TI, uma vez que a micro-segmentação do Data Center ajuda a limitar a vulnerabilidade a ataques e invasões.

1.4. Estimativa de Recursos

Para a implementação da aplicação na infraestrutura da GIRO, será necessário virtualizar os servidores e definir o data center por software. A infraestrutura virtualizada será definida de acordo com as opções do mercado que forem mais adequadas para estrutura definida pela GIRO Telecom. Fatores como preço, segurança e disponibilidade são fundamentais para a escolha.

Logo uma estimativa básica de ferramentas e recursos necessários para a implementação da aplicação é:

- **Sistema de Virtualização ou Serviço de Nuvem Terceirizado**
Sistemas de Virtualização como os oferecidos pela VMWARE ou mesmo serviços já consolidados como a nuvem da AMAZON.
- **Servidor Virtualizado para o FrontEnd da Aplicação**
Neste servidor será instalado o apache web server e os arquivos da aplicação phpIPAM
- **Servidor Virtualizado para o Banco de Dados como BackEnd**
Neste servidor será instalado o servidor de banco de dados MySQL, que armazenará todo o conteúdo gerado na aplicação phpIPAM.
- **Requerimento Mínimo de Recursos para os Servidores**
Cada Servidor precisará dos seguintes elementos de hardware: CPU: DualCore 2.4Ghz; RAM: 4096 MB; Disco: SAS 1024GB; Rede: 2 interfaces Gigabit Ethernet

1.5. Plano de Implantação

Alguns passos podem ser listados no planejamento da implantação do serviço, são eles:

1. Aquisição de Serviço/Sistema de Virtualização

- Para a base do modelo SDDC é necessário um sistema de virtualização para implementar os serviços fundamentais para a aplicação. O tempo estimado para a aquisição considera a cotação de orçamentos do sistema e equipamentos necessários para a implementação. **Prazo: 15 Dias**

2. Instalar/Configurar Infraestrutura de Redes

- Definido o serviço de virtualização será necessário criar a infraestrutura de redes, mesmo que virtual, para atender a comunicação entre os servidores e o acesso dos usuários à aplicação. **Prazo: 1 Dia – Tempo estimado 2 horas**

3. Instalar Servidor de Banco de Dados

- Instalar 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 nomeado *phpipam*
- Liberar permissão de acesso ao banco para o servidor WEB

Prazo: 1 Dia – Tempo estimado 2 horas

4. Instalar Servidor WEB

- Instalar sistema operacional *Ubuntu Server 16.04*
- Configurar 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*)

- Configurar um Virtualhost do servidor WEB para refletir o diretório da aplicação em /var/www/html/phpipam

Prazo: 1 Dia – Tempo estimado 2 horas

5. Instalar a Aplicação

- Clonar código da aplicação pelo Github no diretório /opt do servidor WEB Apache
- Editar a configuração no arquivo *config.php* da aplicação de acordo com as informações do banco de dados e o diretório do *Virtualhost*
- Habilitar e configurar módulo *rewrite* no servidor WEB
- Acessar a interface WEB do phpIPAM para instalar a base de dados padrão da aplicação no servidor BackEnd
- Definir método de login e registro de usuários na plataforma

Prazo: 1 Dia – Tempo estimado 2 horas

6. Homologação da Aplicação

- Testes da aplicação e interação entre os servidores
- Avaliação da *Interface de Usuário* e usabilidade do sistema
- Testes de adição, remoção, manipulação e edição das sub-redes
- Avaliar recursos extras da aplicação
- Realizar testes com os usuários
- Adicionar redes, sub-redes e sistemas autônomos

Prazo: 10 Dias

	DIAS																													
ATIVIDADES	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1. Aquisição																														
2. Instalar e Configurar Rede																														
3. Instalar Banco de Dados																														
4. Instalar Servidor WEB																														
5. Instalar a Aplicação																														
7. Homologar Aplicação																														

Cronograma de Implantação

Com os testes de homologação realizados, a aplicação já estará pronta para entrar em produção.

1.6. Definições de Rede

Para o funcionamento pleno dos serviços distribuídos da aplicação uma infraestrutura de redes deve ser definida. Independente de qual infraestrutura será utilizada para implementar a aplicação.

Para o servidor phpIPAM serão necessárias duas interfaces de rede, uma conectada à rede pública e outra em uma rede interna privada, que fará a comunicação com o servidor de banco de dados MySQL, que por sua vez terá uma interface de rede apenas e que estará conectada à rede privada mencionada anteriormente.

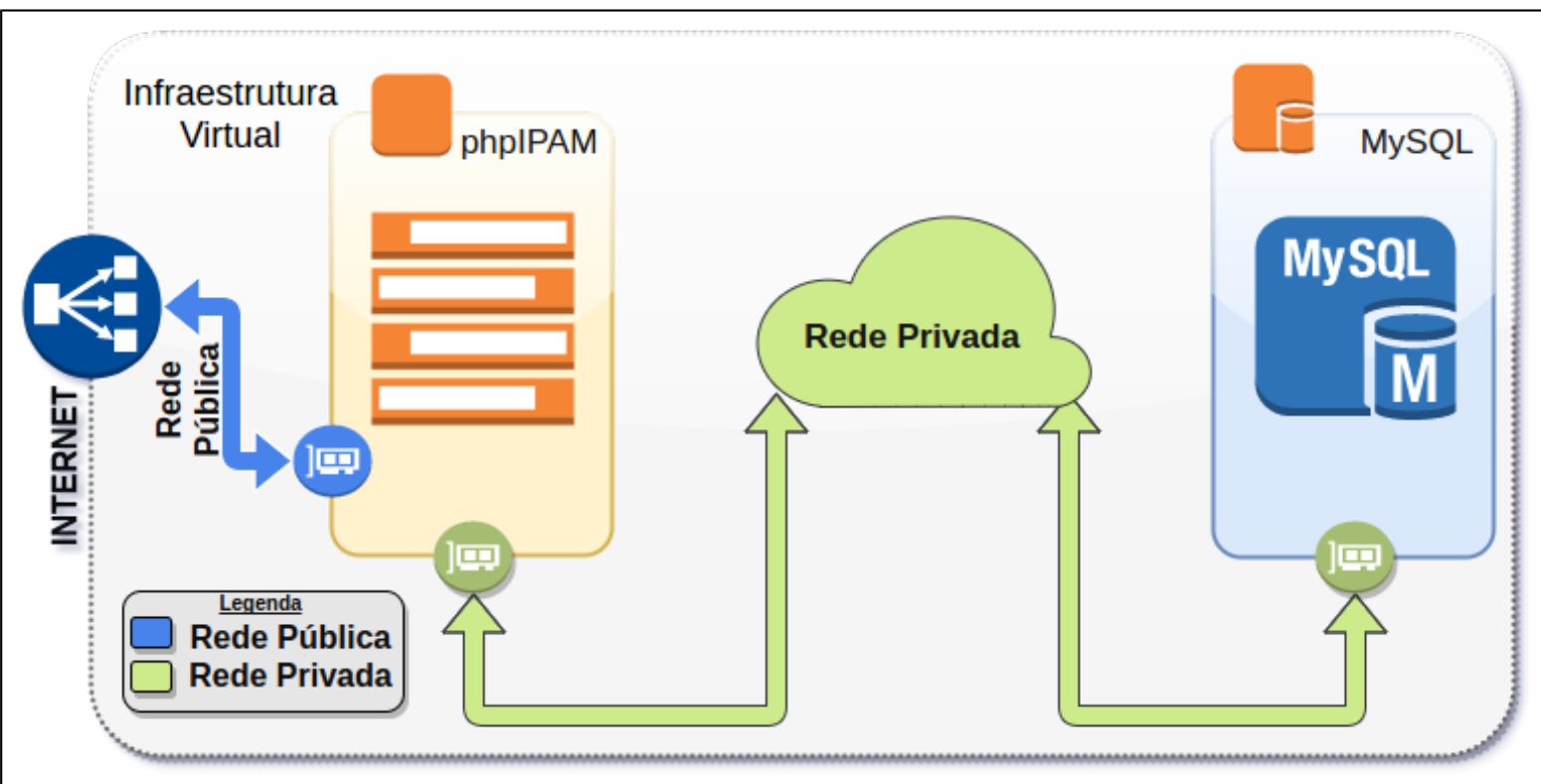


Figura 3. Diagrama de Rede da infraestrutura phpIPAM

Implementando uma infraestrutura de redes como no diagrama acima, a aplicação será executada corretamente e o banco de dados estará protegido em uma rede segmentada apenas. Nesta rede apenas as informações de banco de dados serão trocadas entre os servidores.

1.7. Alternativas de Implementação

A implementação de aplicações em nuvem podem ser muito simples e eficientes, se considerar que infraestruturas completas disponíveis no mercado como a AWS da Amazon ou mesmo o framework OpenStack.

Abaixo são apresentados ambos os modelos para a implementação da aplicação em nuvem utilizando os conceitos do **OpenStack e AWS**.

1.7.1. Exemplo AWS

A Amazon Web Services (AWS) é uma plataforma de serviços em nuvem, que oferece armazenamento de banco de dados, distribuição de conteúdo, poder de computacional e outras funcionalidades que ajudam as empresas no dimensionamento e crescimento de seus negócios construindo aplicações sofisticadas com mais flexibilidade, escalabilidade e confiabilidade.

Baseado nos diagramas preparados para a implementação do phpIPAM na infraestrutura da GIRO Telecom, um desenho da infraestrutura utilizando os serviços da Amazon foi criado para ilustrar uma implementação utilizando essa plataforma.

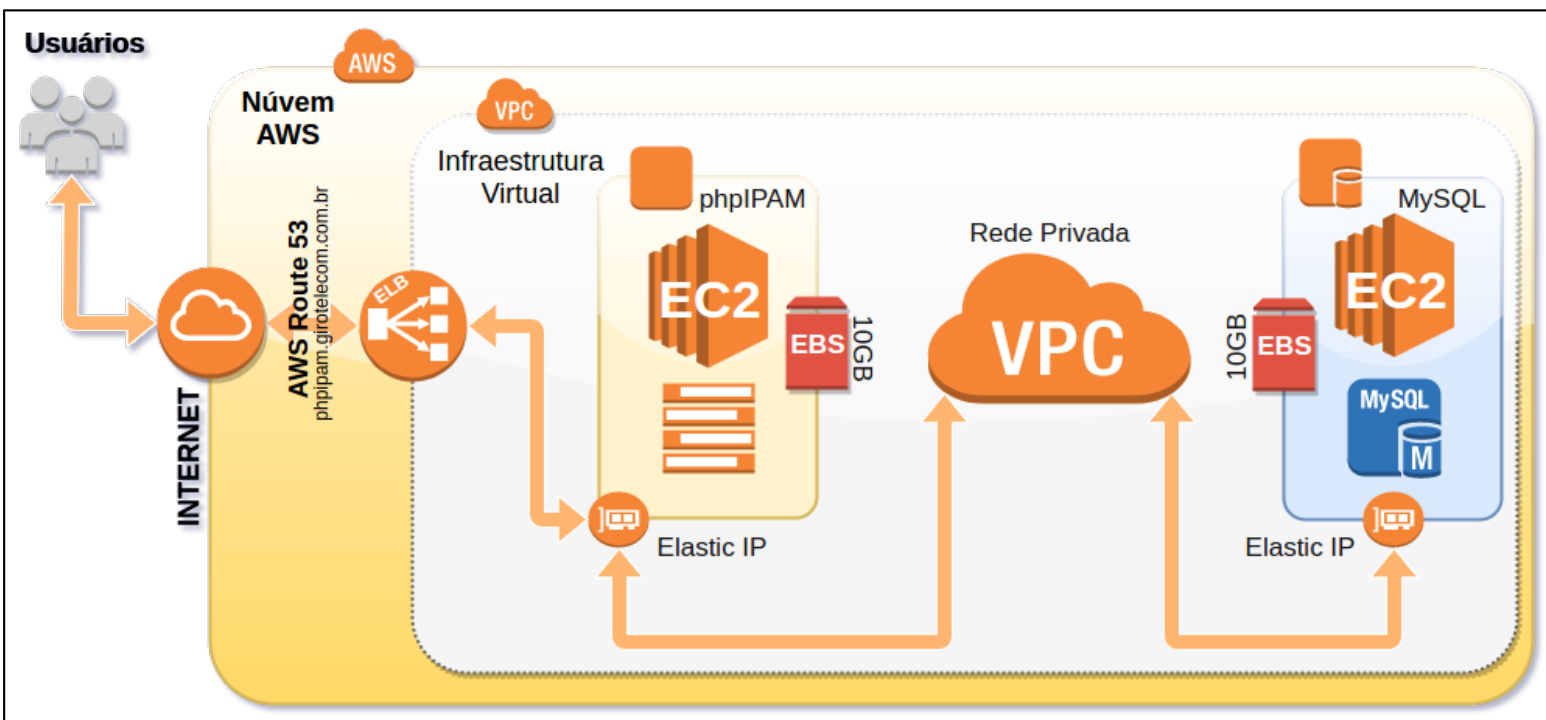


Figura 4. phpIPAM na plataforma AWS

Analisando o diagrama da figura 4, é possível destacar os serviços AWS necessários para a implementação do phpIPAM na nuvem da Amazon.

Os usuários, através de um navegador WEB, acessam a nuvem AWS para alcançar a aplicação utilizando o hostname definido na zona girotelecom.com.br, hospedada no serviço **AWS Route 53**. O tráfego é balanceado e controlado pelo serviço **Elastic Load Balance**, que busca trafegar os dados com a melhor qualidade possível e menor latência dentro da nuvem. Em seguida o recurso **Elastic IP** cria a comunicação do servidor isolado para o mundo além de garantir a disponibilidade dos serviços independentes da região em que são acessados. O Front-end da aplicação exibido com o servidor HTTP Apache instalado em uma instância **EC2** (*Elastic Compute Cloud*) com o SO *Ubuntu Server 16.04*, tem a aplicação phpIPAM armazenada no volume de armazenamento **EBS** (*Elastic Block Storage*), executa a homepage da aplicação, lendo e gravando as informações na outra instância **EC2**, **também** com o SO *Ubuntu Server 16.04*, onde roda o Servidor de banco de dados MySQL e tem seu próprio volume **EBS**. Essa comunicação entre as instâncias é realizada através da rede virtual provida pelo serviço **VPC** simples, utilizando o protocolo mysql. Tanto a instância do phpIPAM quanto do MySQL possuem um volume **EBS** de 10GB de disco e um sistema de *snapshots* diários para maior segurança dos dados, mas essa capacidade pode ser alterada caso necessário.

1.7.2. Exemplo OpenStack

OpenStack é um software Open Source pode ser instalado preferencialmente em vários servidores agregando uma camada de orquestração para o gerenciamento da plataforma e seus componentes.

Cada um desses servidores será responsável por um ou mais serviços do OpenStack, uma vez que a plataforma é totalmente modular.

Abaixo um diagrama exemplificando a implementação da infraestrutura do phpIPAM em um ambiente OpenStack, seguindo os mesmos conceitos apresentados nos diagramas anteriores.

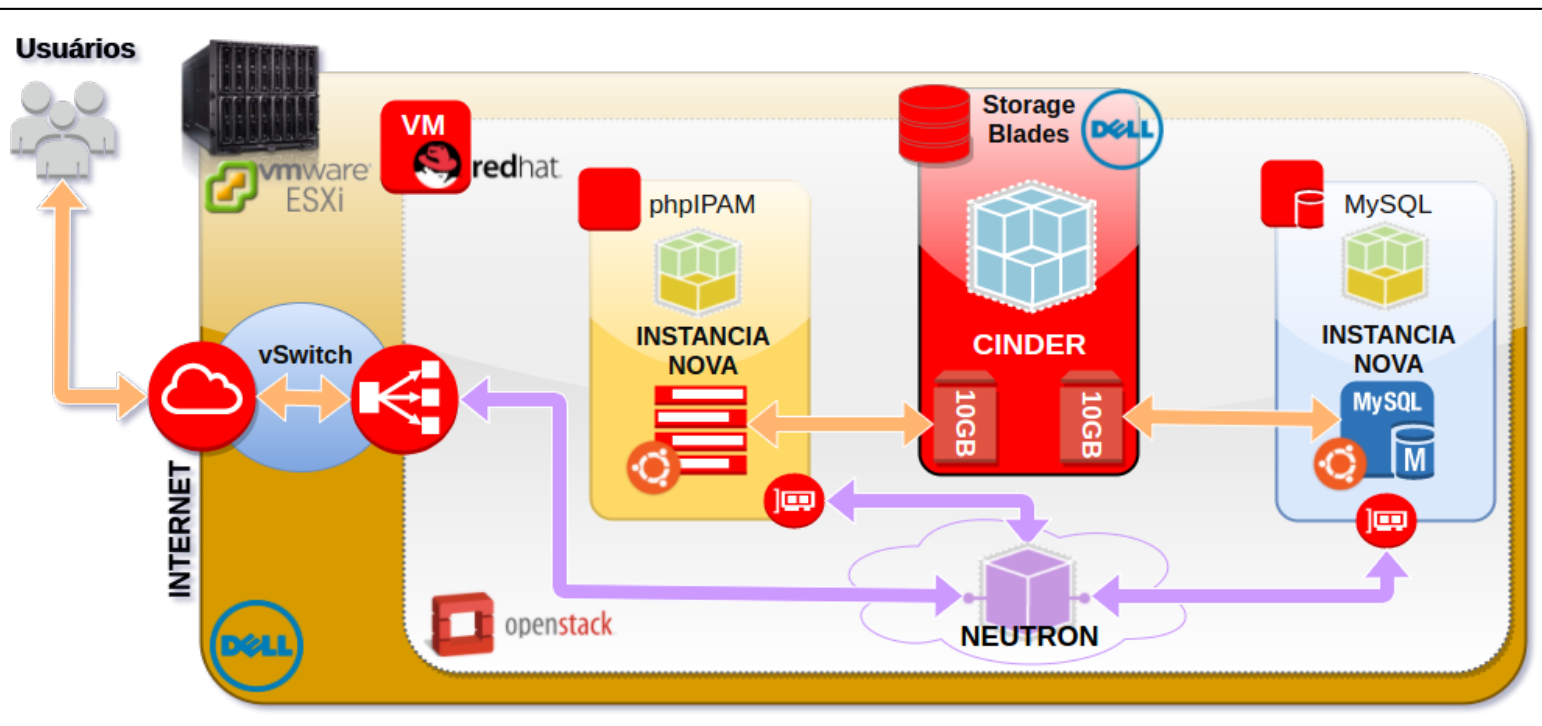


Figura 5. phpIPAM na plataforma OpenStack

Analisando o diagrama da figura 5, é possível destacar os serviços OpenStack necessários para a implementação do phpIPAM na nuvem da privada.

Os usuários, através de um navegador WEB, acessam a nuvem OpenStack para alcançar a aplicação utilizando o hostname definido na zona girotelecom.com.br, hospedada no serviço **DNS Registro.BR**. O tráfego é balanceado e controlado pelo serviço **NEUTRON**, integrado com a interface **VSS** do **ESXi**, que busca trafegar os dados com a melhor qualidade possível e menor latência dentro da nuvem. Em seguida o recurso **NEUTRON** cria a comunicação do servidor isolado para o mundo, sendo possível garantir a disponibilidade dos serviços de rede. O Front-end da aplicação exibido com o servidor **HTTP Apache 2** instalado em uma instância **NOVA** com o SO *Ubuntu Server 16.04* sob o **hypervisor QEMU**(suportado pelo OpenStack), tem a aplicação phpIPAM armazenada no volume de armazenamento de Bloco **CINDER**, que controla o *volume driver Dell EqualLogic*, executa a homepage da aplicação, lendo e gravando as informações na outra instância **NOVA com hypervisor QEMU**, também com o SO *Ubuntu Server 16.04*, onde roda o servidor de banco de dados **MySQL 5.6** e tem seu próprio volume no **CINDER**. Essa comunicação entre as instâncias é realizada através da rede virtual

provida pelo serviço **NEUTRON**, ambas utilizando o protocolo **mysql** uma como cliente e outra como servidor. Tanto a instância do phpIPAM quanto do MySQL possuem um volume **CINDER** de 10GB de disco e um sistema de *snapshots* diários para maior segurança dos dados, mas essa capacidade pode ser alterada caso necessário.

Toda a infraestrutura de nuvem privada foi instalada em um *Cloud Server Dell M1000E* utilizando o **VMWARE ESXi**. O OpenStack instalado em uma instância virtual em cluster no **SO RedHat**.

1.7.3. Exemplo vSphere

O VMware vSphere é uma plataforma de virtualização que simplifica a TI segmentando aplicativos e sistemas operacionais do hardware subjacente. Os seus dispõem de recursos dedicados, mas ainda assim os servidores podem ser gerenciados como um *pool* de recursos. Dessa maneira, um ambiente de TI pode ser simplificado, mas resiliente.

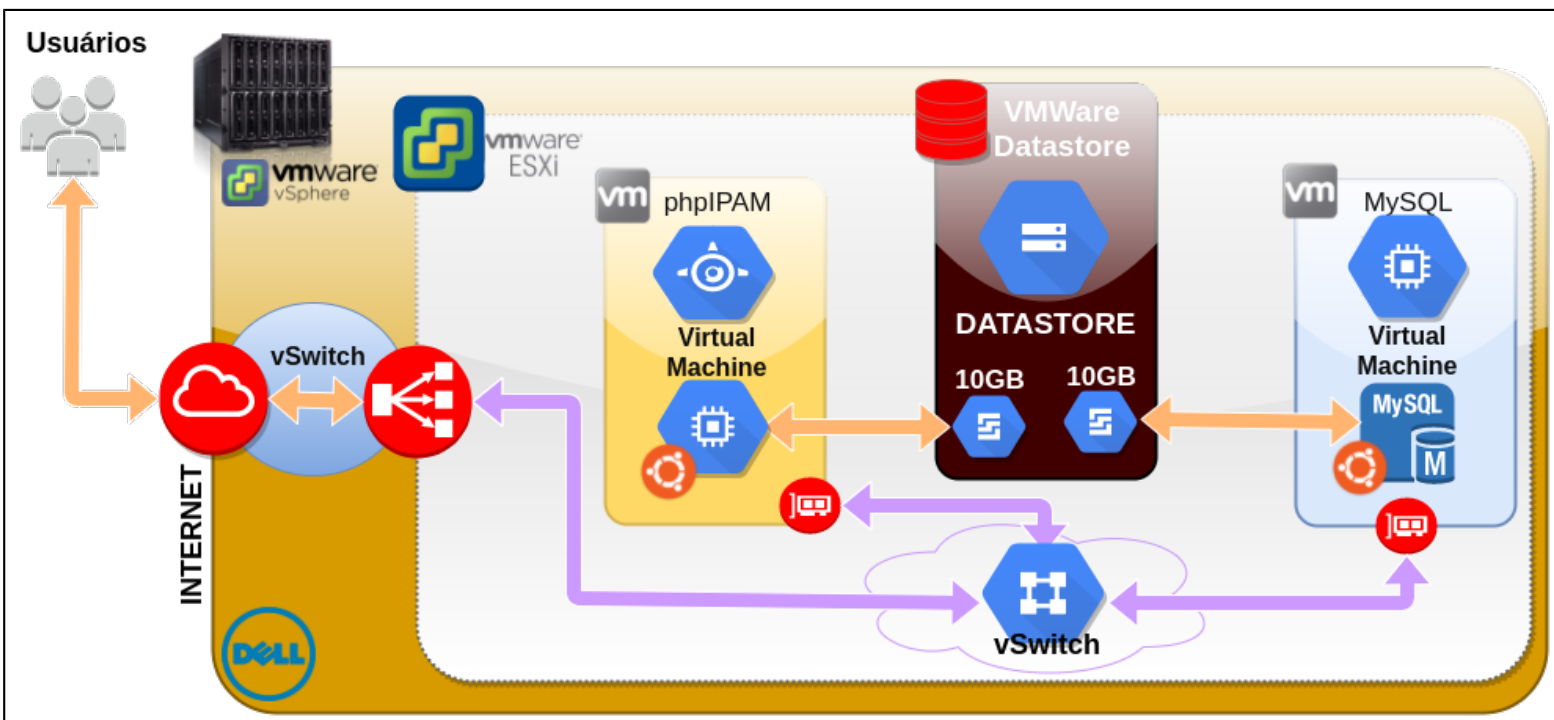


Figura 6. phpIPAM na plataforma VMWare vSphere

Os usuários, a partir de um navegador WEB, podem acessar a aplicação hospedada na plataforma **vSphere** como uma instância virtual. Em uma segunda instância existe um servidor de banco de dados que trabalha em

conjunto com a aplicação. A infraestrutura de redes no vSphere é realizada através do recurso **vSwitch** que interliga as instâncias virtuais e a rede física fora do hypervisor. O armazenamento de disco é feito com o gerenciador de armazenamento, ou **datastore**, recurso responsável por gerenciar os discos virtuais e armazenamento em geral. O **VMware ESXi** é o *hypervisor* instalado diretamente no servidor físico, podendo ser dividido em diversos servidores lógicos chamados de máquinas virtuais. É neste *hypervisor* que as instâncias virtuais dos servidores da aplicação estão instalados. O **vSphere** gerencia múltiplos servidores **ESXi**, como data centers virtuais onde é possível ter diversas máquinas virtuais, como o caso da aplicação phpIPAM.

Toda a infraestrutura de foi instalada em um *Cloud Server Dell M1000E* utilizando o **VMWARE ESXi e VMWARE vSphere**. O phpIPAM instalado em duas instâncias virtuais em cluster com o **SO Ubuntu Server**.

1.7.4. Analisando as Opções

Analisando as opções demonstradas, o que fica mais próximo do cenário apresentado seria uma solução de nuvem privada como apresentado no modelo OpenStack. A aplicação phpIPAM foi apresentada neste documento em uma infraestrutura bem simples, que poderia ser de fácil implementação na plataforma AWS, até mais barato inclusive. Mas seguindo a proposta apresentada no início, a aquisição dos equipamentos necessários para a implementação da infraestrutura, combinado com as possibilidades de ter uma plataforma própria de virtualização com infraestrutura em nuvem, é bem interessante para a Giro Telecomunicações ter investimentos nessa área para negócios futuros. Logo a implementação do serviço com **VMWARE ESXi** poderia ser o próximo passo da empresa para criação de novos serviços.

Em primeira instância, a implantação do phpIPAM será realizada no ambiente virtualizado *vSphere* da **VMWARE** o que poderá ser migrado posteriormente segundo a necessidade de escalonamento da aplicação.

1.8. Preparando a Solução em Nuvem

A implementação do ambiente vSphere demonstrada a seguir é apenas um exemplo de como devem ser executados os passos. Nesse procedimento foi realizada a instalação do *Hypervisor* em um ambiente virtualizado, mas o mesmo pode ser executado em um ambiente físico.

A arquitetura de *hypervisor* ESXi foi instalado em um servidor com os requerimentos apresentados no item **1.4**, seguindo os seguintes passos. A versão utilizada na instalação é *VMware ESXi 6.5.0*.

1.8.1. Iniciando a instalação

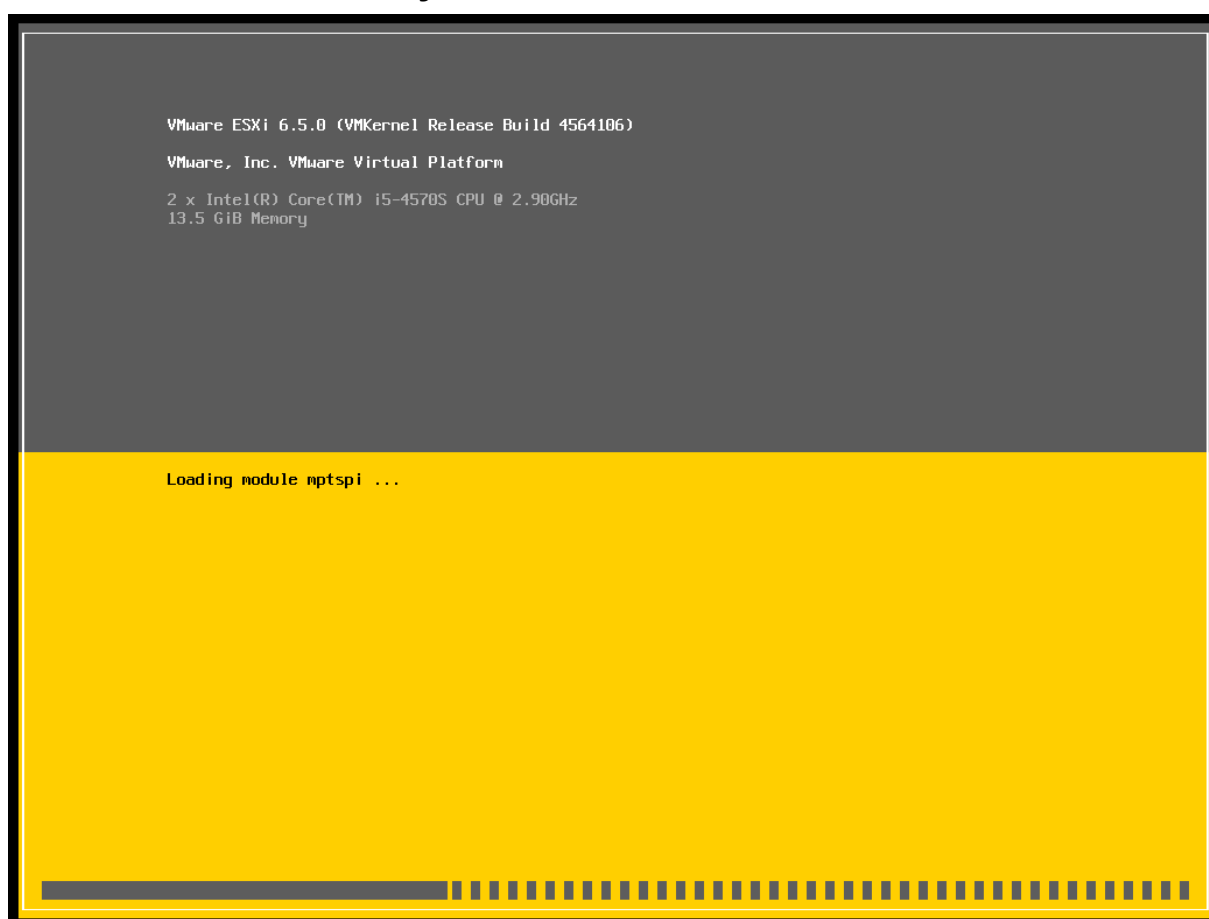
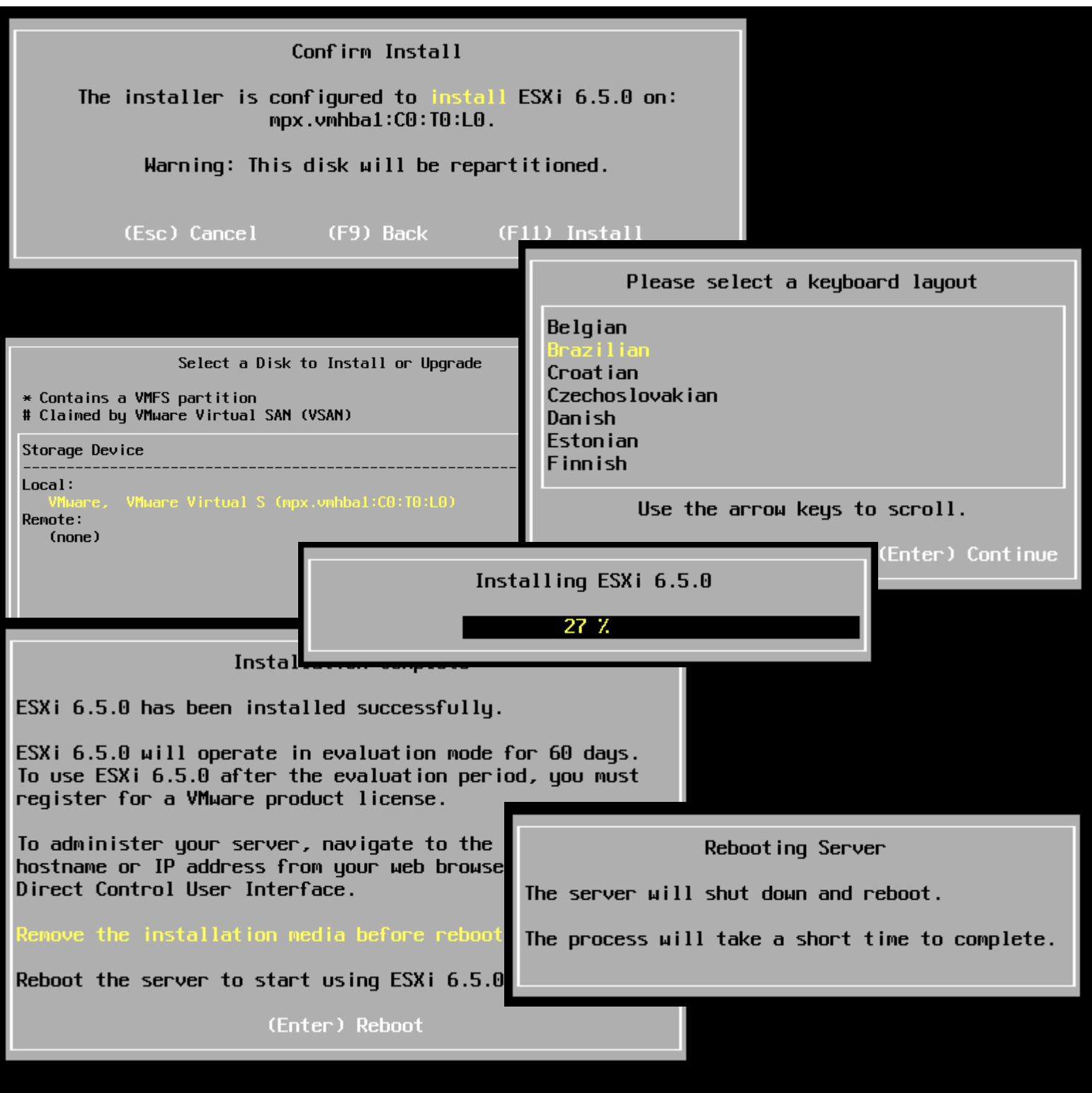


Figura 7. Implementando o ESXi

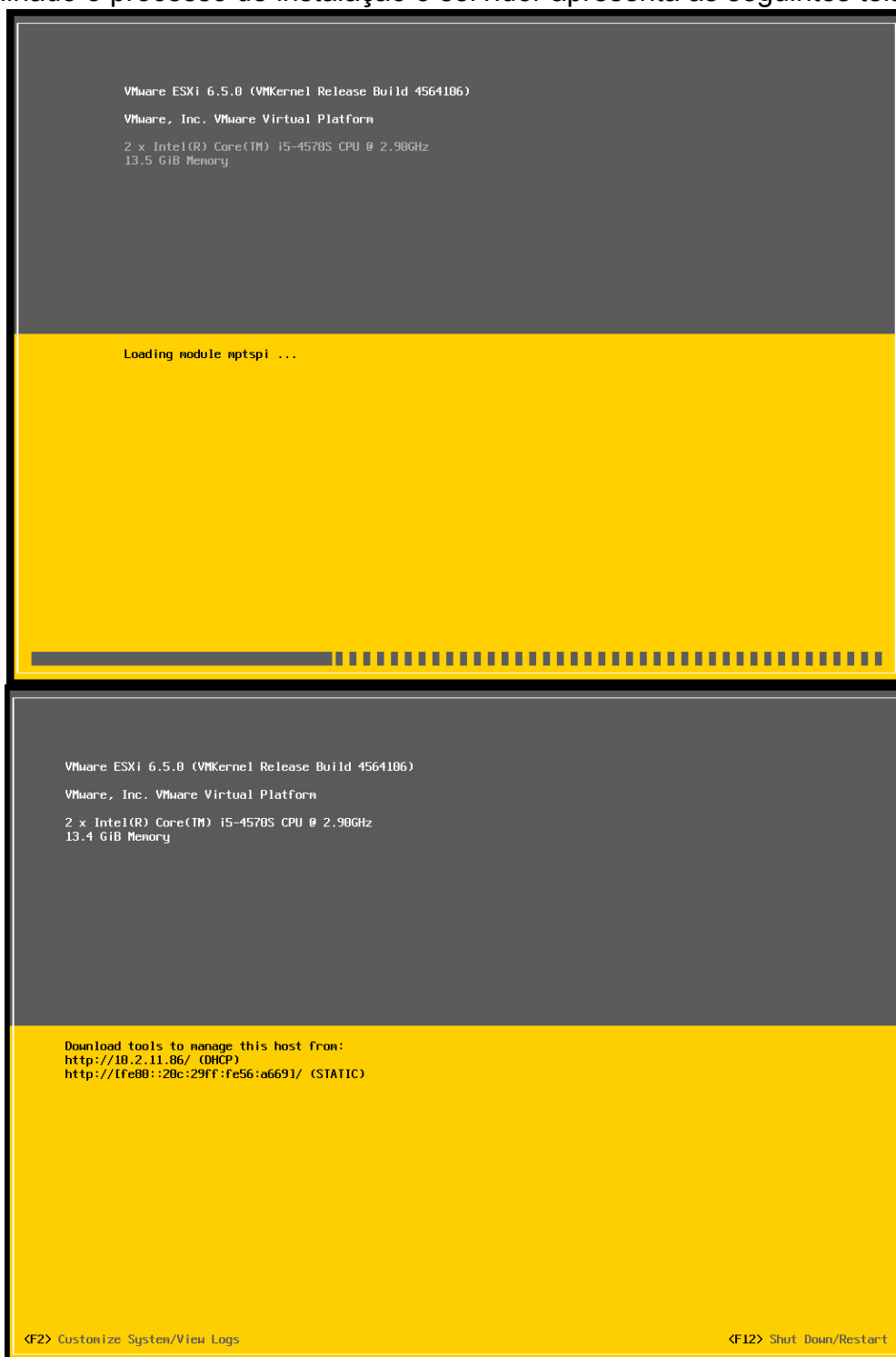
O quadro acima demonstra a tela de inicialização do processo de instalação do ESXi. O quadro abaixo demonstra os momentos iniciais da instalação do ESXi como a confirmação de instalação, seleção do Layout do teclado e finalização.



Quadro 1. Momentos da instalação do ESXi

1.8.2. Iniciando Servidor ESXi

Terminado o processo de instalação o servidor apresenta as seguintes telas:



Quadro 2. ESXi Inicializando

Com a instalação concluída, basta acessar a interface WEB do servidor com o IP indicado na imagem acima para iniciar a instalação do servidor Windows.

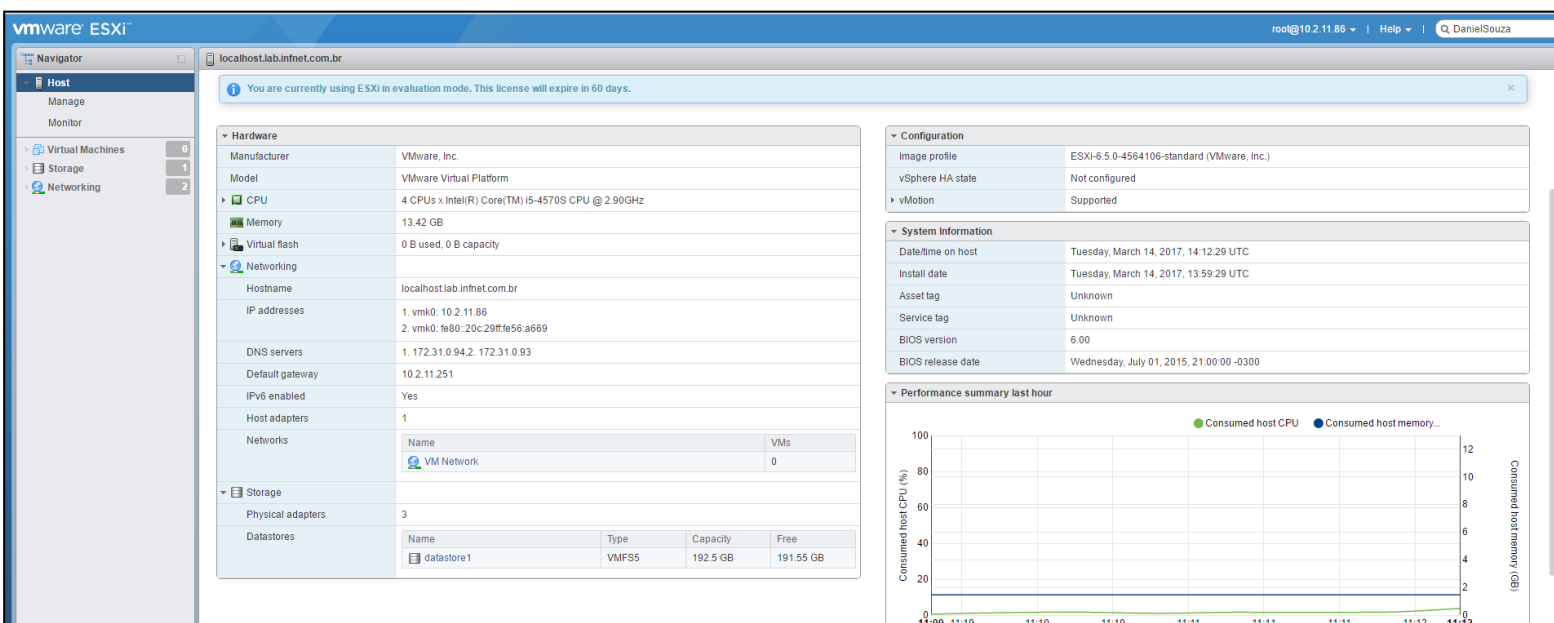


Figura 8. Interface WEB do ESXi

Hardware

Manufacturer

VMware, Inc.

Model

VMware Virtual Platform

CPU

4 CPUs x Intel(R) Core(TM) i5-4570S CPU @ 2.90GHz

Memory

13.42 GB

Virtual flash

0 B used, 0 B capacity

Networking

Hostname

localhost.lab.infnet.com.br

IP addresses

1. vmk0: 10.2.11.86
2. vmk0: fe80::20c:29ff:fe56:a669

DNS servers

1. 172.31.0.94, 2. 172.31.0.93

Default gateway

10.2.11.251


IPv6 enabled

Yes

Host adapters

1

Networks

Name	VMs
 VM Network	0

Storage

Physical adapters

3

Datastores


Name	Type	Capacity	Free
 datastore1	VMFS5	192.5 GB	191.55 GB

Figura 9. Confirmando Configuração Instalada

1.9. Instalando vCenter

A instalação do vCenter depende apenas da criação da máquina virtual onde será instalado o *Windows Server 2012*(figura 10.) e em seguida a instalação do próprio sistema operacional em si.

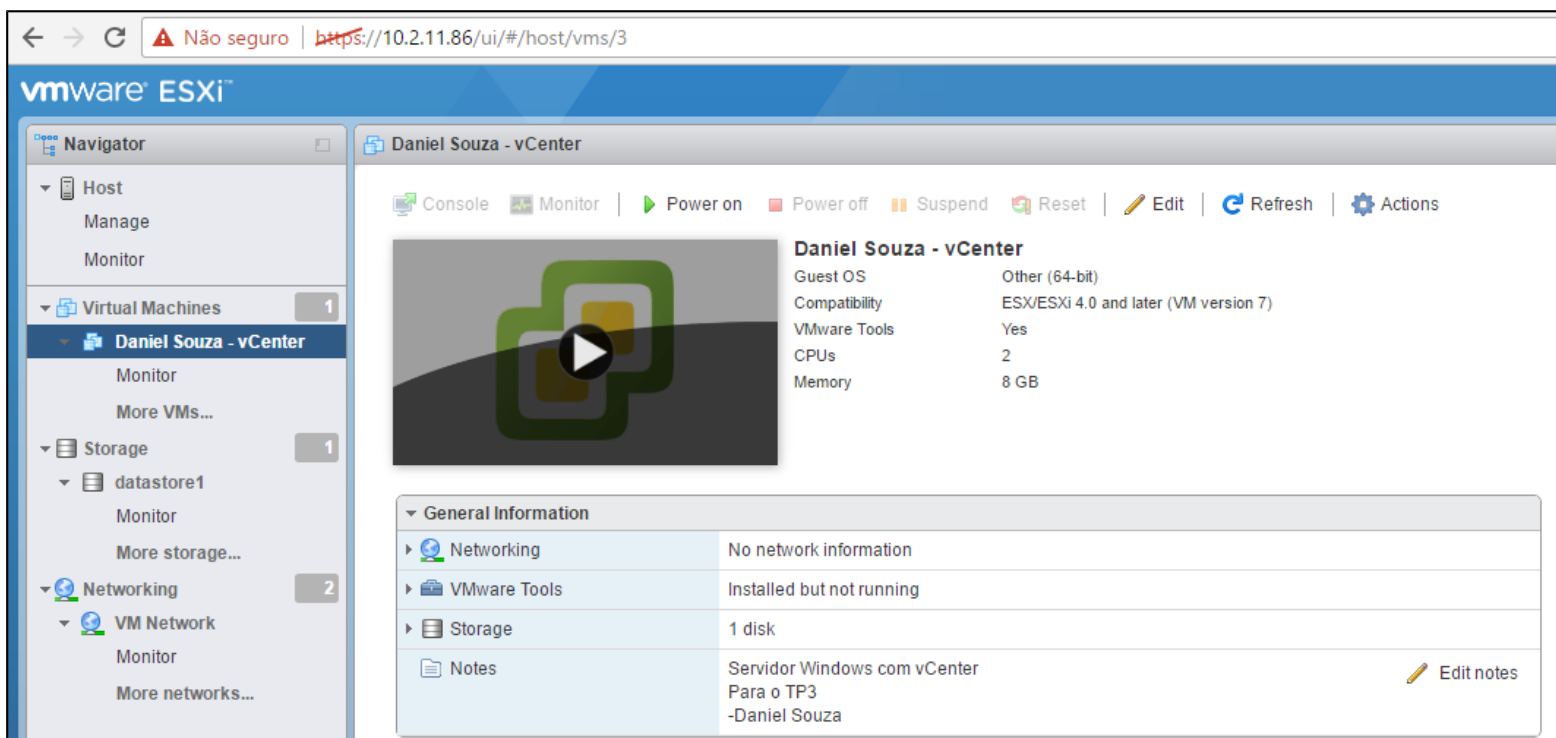


Figura 10. Criação da Máquina Virtual para o Windows Server

1.9.1. Resolvendo dependências

Para que o vCenter possa ser instalado corretamente é necessário cumprir algumas dependências. Segundo a documentação da aplicação encontrada do site oficial(<http://www.vmware.com/br/support/vcenter-server.html>), é preciso que haja um *FQDN*(*Fully Qualified Domain Name*), isto é, um nome de domínio designado ao endereço IP do servidor. Isto se dá pois o acesso a interface WEB do vCenter utiliza um *virtualhost* baseado no *FQDN* designado para o servidor onde foi instalado.

Sabendo da dependência de um *FQDN* para o servidor, é necessário que um serviço de *DNS* seja utilizado para a criação de uma zona de domínio para a criação do *FQDN* do servidor. Infelizmente o vCenter não consegue ser

instalado em um servidor que tenha algumas portas ocupadas(figura 11), entre elas a 53 designada para DNS.


 The following ports must also be available for this deployment:
53, 88, 389, 636, 2012, 2014, 2015, 2020, 5480, 7080, 7081, 8200, 8201, 8300, 8301, 11711, 11712, 12721

Figura 11. Portas que devem estar liberadas para o vCenter

Para a instalação do vCenter é necessário que todas as portas listadas a cima estejam liberadas no servidor em todas as suas interfaces, sejam elas físicas ou virtuais.

Com esta limitação em mente, foi importado para o ESXi um arquivo *OVF* de um servidor Linux com o serviço DNS devidamente instalado e nele foi configurada a zona de domínio *danielsouza.edu.br*. Nesta zona foi adicionado o *FQDN vcenter.danielsouza.edu.br* apontando para o IP do servidor Windows 2012, que também foi importado com um Arquivo *OVF* para agilizar a instalação.

Com o servidor DNS devidamente configurado, tendo uma zona definida para o servidor windows, instalação do vCenter poderá em fim dar início.

1.9.2. Iniciando a instalação do vCenter

O DVD de instalação do vCenter contém um *setup* que, se executado, inicia a instalação do vCenter. As próximas imagens mostram alguns momentos da instalação. O software utilizado nesta instalação foi o *vCenter Server 6.5.0*.

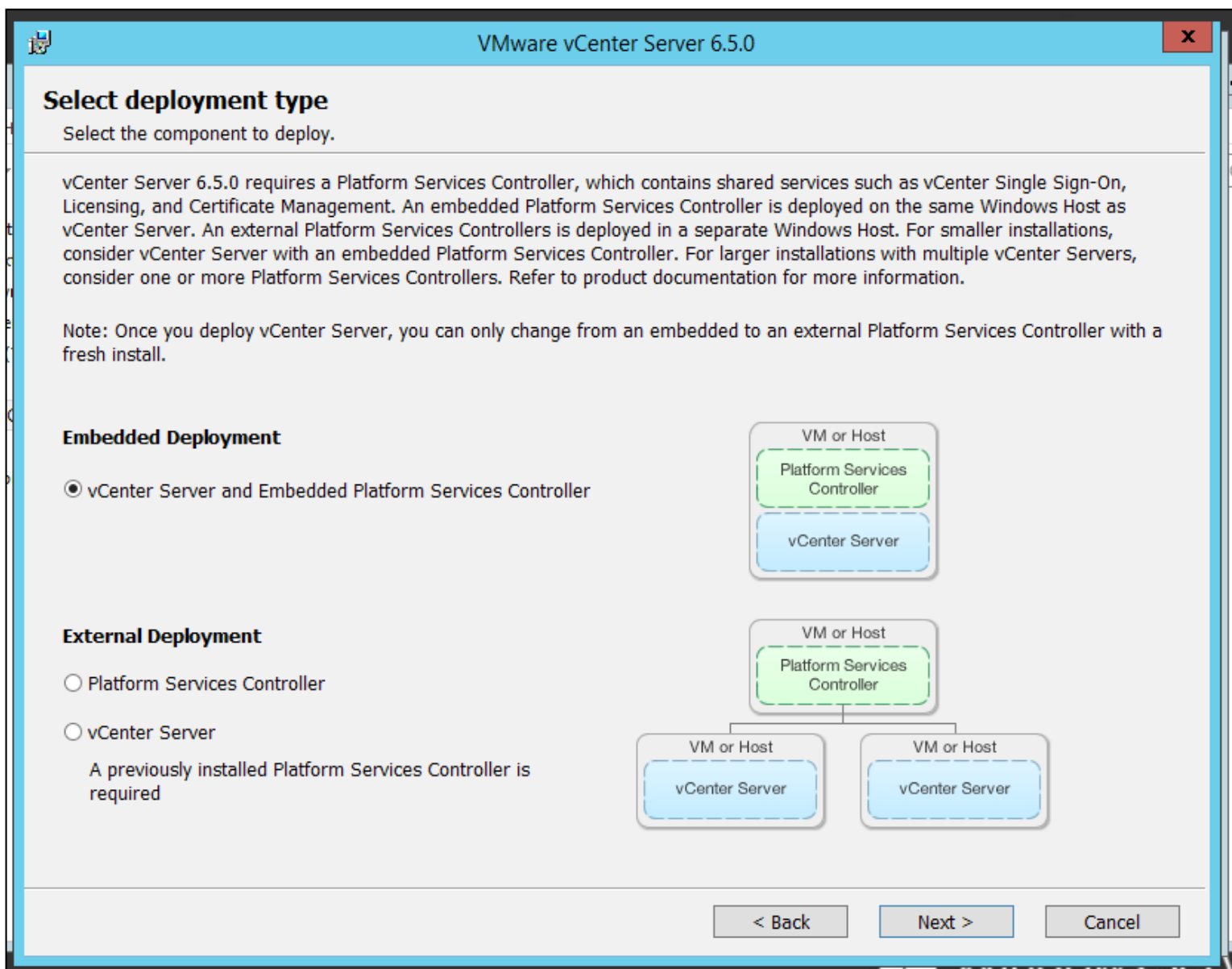
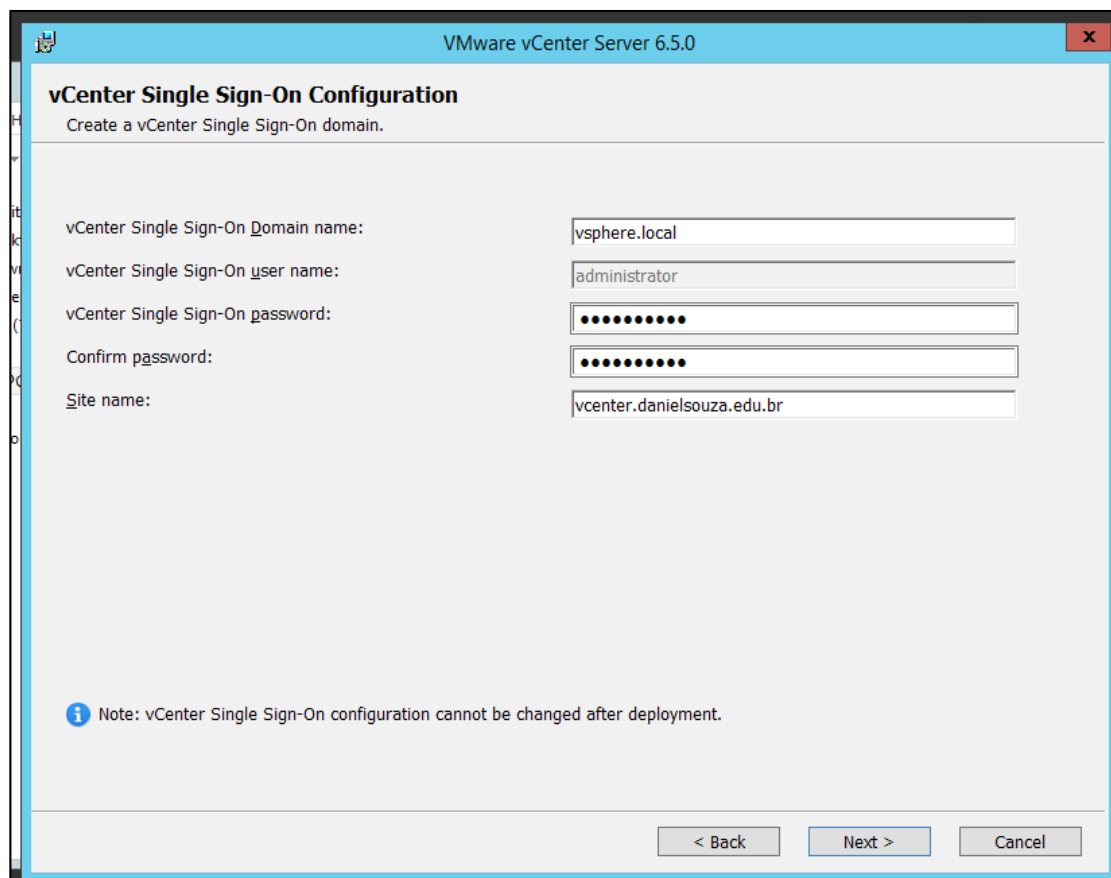


Figura 12. Iniciando o Setup de instalação



VMware vCenter Server 6.5.0

vCenter Single Sign-On Configuration
Create a vCenter Single Sign-On domain.

vCenter Single Sign-On Domain name: vsphere.local

vCenter Single Sign-On user name: administrator

vCenter Single Sign-On password:

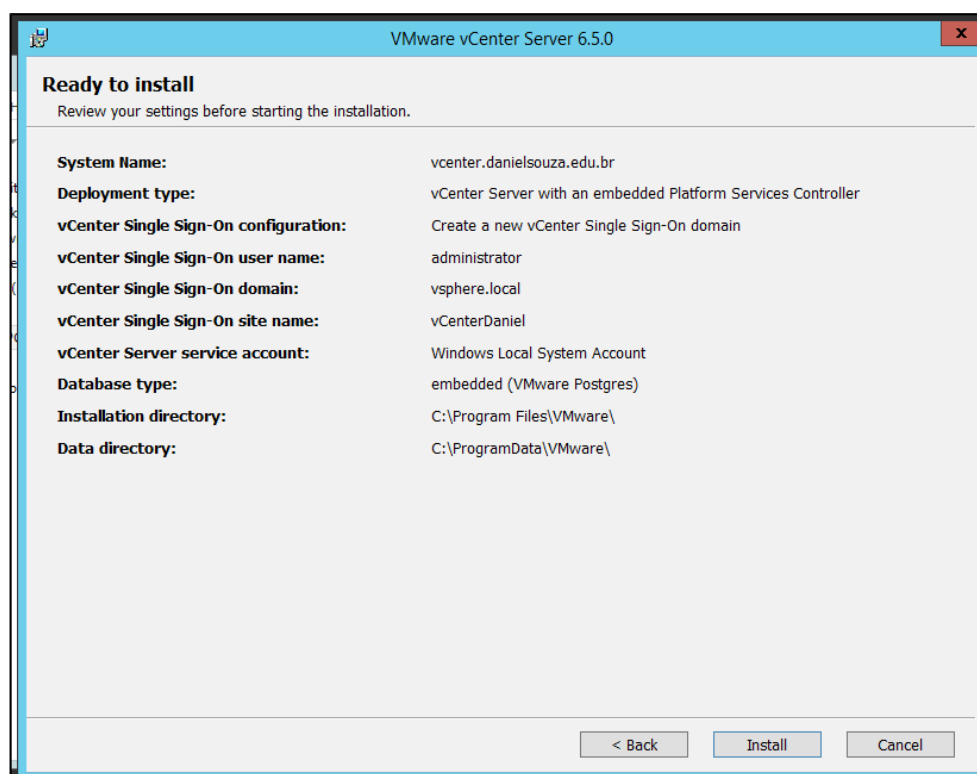
Confirm password:

Site name: vcenter.danielsouza.edu.br

Note: vCenter Single Sign-On configuration cannot be changed after deployment.

< Back Next > Cancel

Figura 13. Configuração de Site Name e Usuário padrão



VMware vCenter Server 6.5.0

Ready to install
Review your settings before starting the installation.

System Name: vcenter.danielsouza.edu.br

Deployment type: vCenter Server with an embedded Platform Services Controller

vCenter Single Sign-On configuration: Create a new vCenter Single Sign-On domain

vCenter Single Sign-On user name: administrator

vCenter Single Sign-On domain: vsphere.local

vCenter Single Sign-On site name: vCenterDaniel

vCenter Server service account: Windows Local System Account

Database type: embedded (VMware Postgres)

Installation directory: C:\Program Files\VMware\

Data directory: C:\ProgramData\VMware\

< Back Install Cancel

Figura 14. vCenter pronto para a instalação

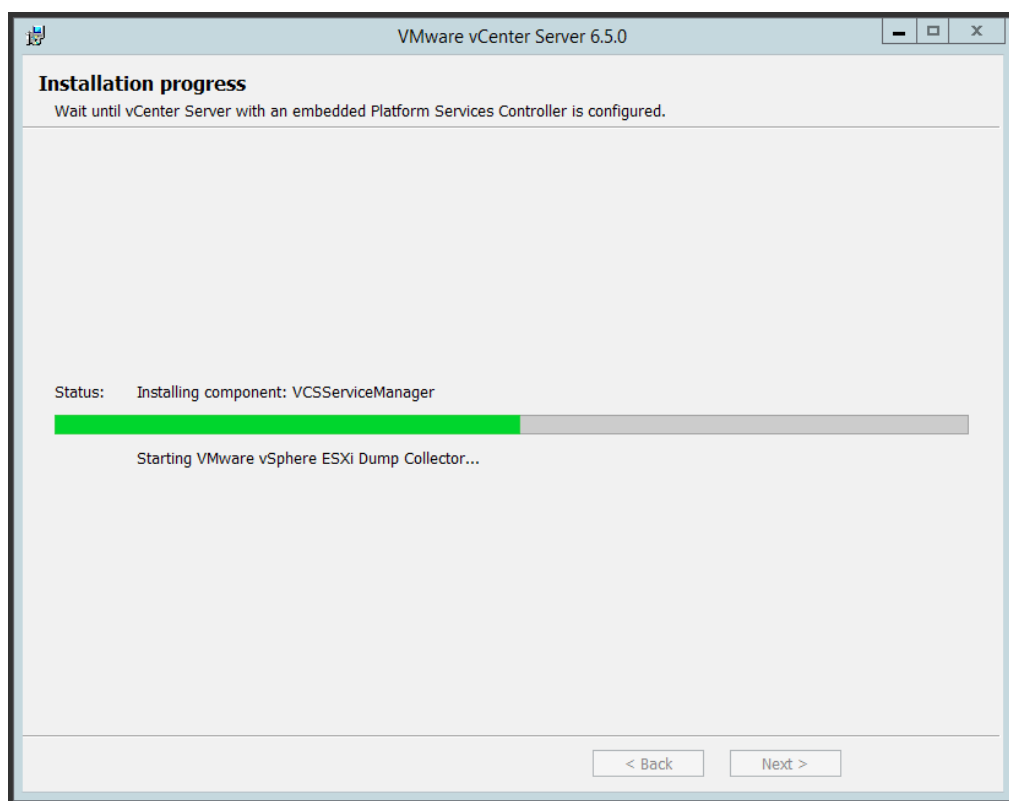


Figura 15. Instalação do vCenter em Andamento

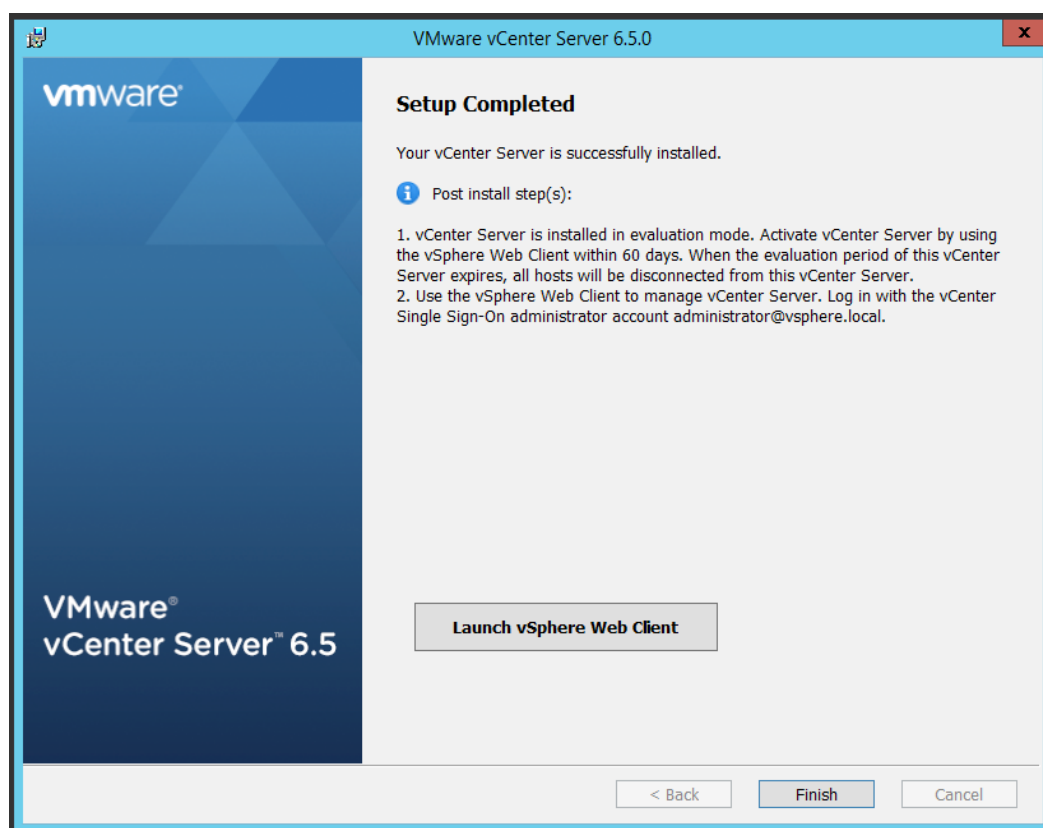


Figura 16. Instalação Concluída

1.9.3. Criando Data Center Virtual.

Após a instalação do vCenter no servidor Windows 2012(figura 17), já é possível realizar a criação do data center virtual.

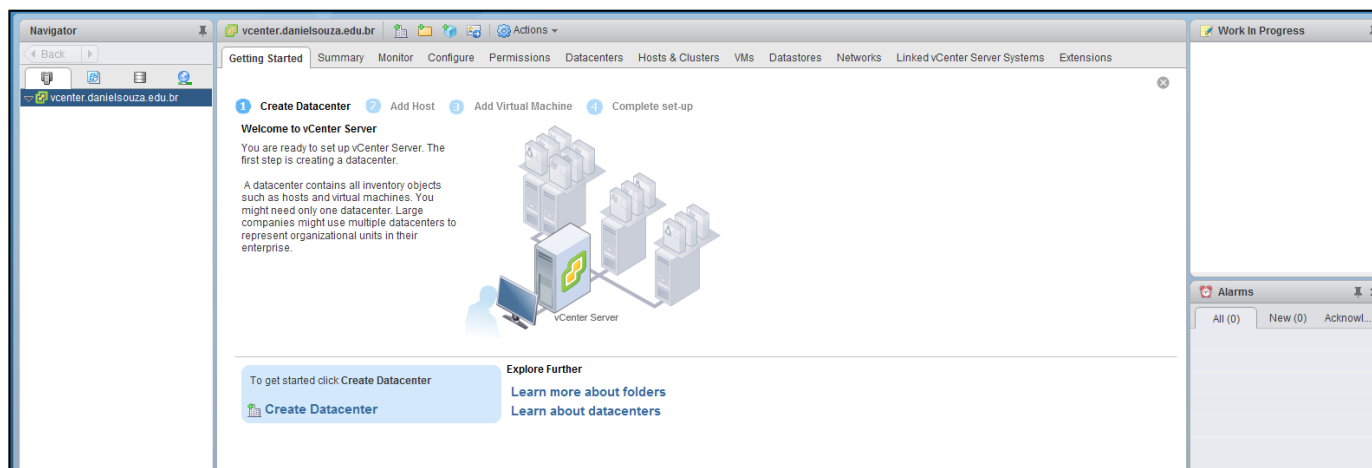


Figura 17. Servidor vCenter/cSphere pela interface WEB

Para esta criação, é necessário primeiro definir o nome do datacenter e em seguida adicionar o *host ESXi*, que foi previamente instalado, clicando em *Add Host* no campo esquerdo da interface Web(figura 18), logo após ter clicado em *Create Datacenter*, onde foi definido o nome DANIEL-DC para o novo datacenter Virtual.

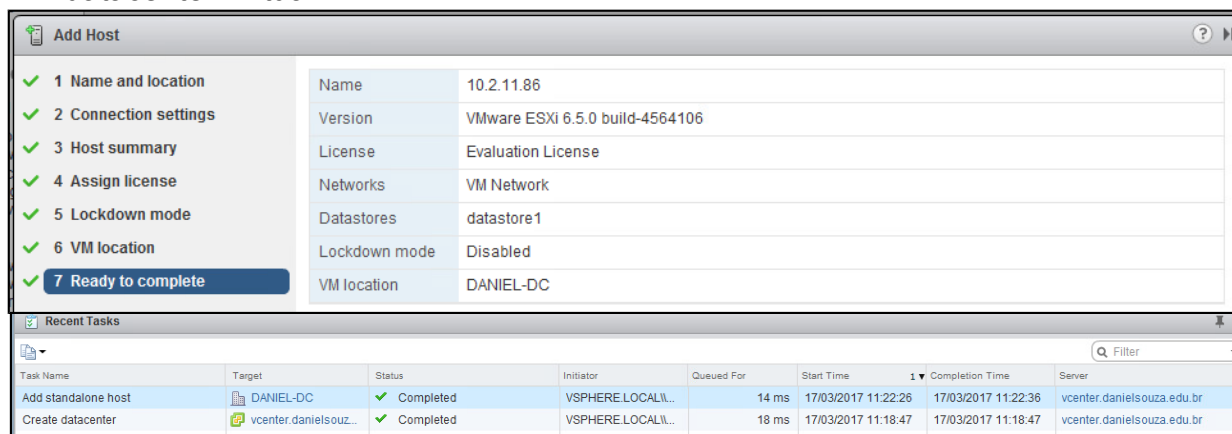


Figura 19. Adicionando Host ESXi no Data Center DANIEL-DC

Feito isto, o novo data center já tem seu primeiro *Host* e está pronto para fazer parte de um novo *cluster* quando *necessário*(figuras 20 e 21).

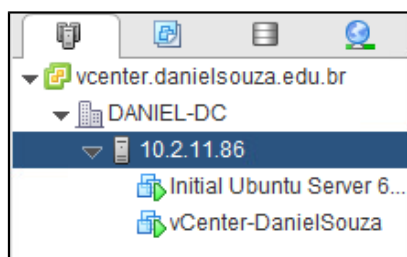


Figura 20. Listagem de inventário

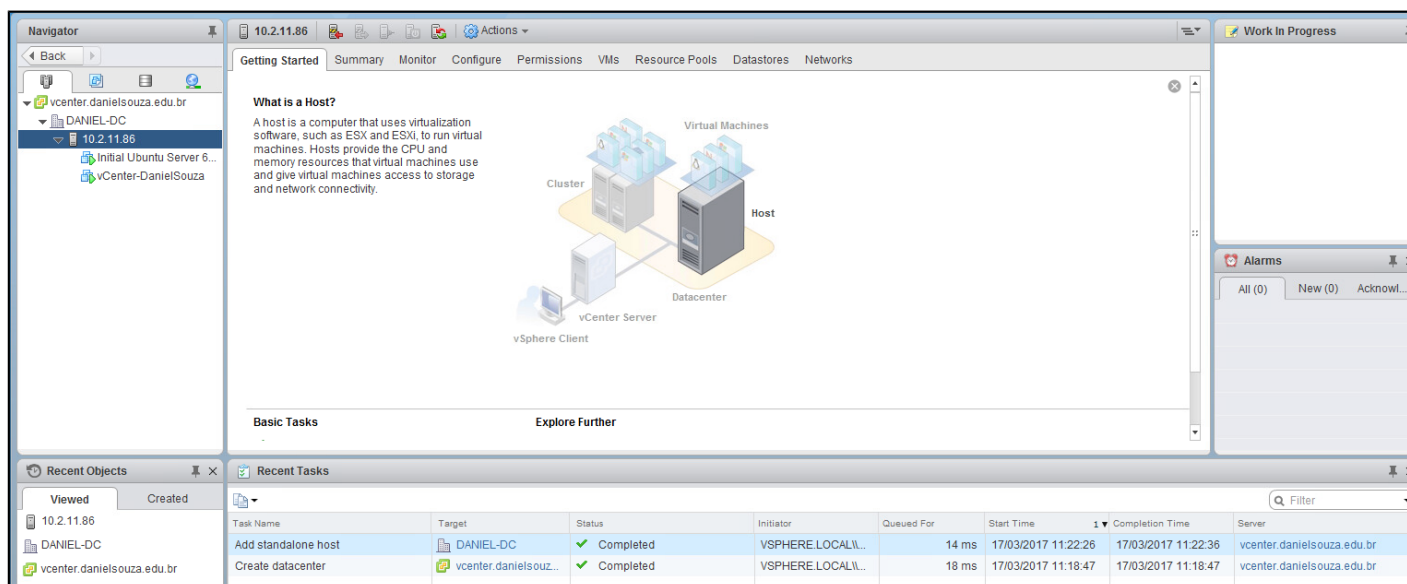


Figura 21. Infraestrutura completa do vCenter com Data Center virtual, seu Host e Máquinas Virtuais.

A figura 21 mostra uma captura Web Client visualizando a infraestrutura completa do vCenter. Nele é possível ver o Datacenter(DANIEL-DC) e o host ESXi com 2 máquinas virtuais, uma é *Initial Ubuntu Server 64-Bit* que roda o serviço de resolução de nomes *bind9* para a configuração do FQDN *vcenter.danielsouza.edu.br* e próprio *Windows Server 2012* contendo a instalação do servidor *vCenter*.

Depois da implementação do ambiente virtualizado, basta criar as duas máquinas virtuais necessárias para a implementação da Aplicação phpIPAM.

Nesta implementação apenas 2 instâncias virtuais serão necessárias. Os requisitos mínimos exigidos são: 1vCPU 64Bit, RAM 1GB, Disco 10GB, uma instância com 2 Interfaces de Rede para o servidor e outra instancia com 1 interface de rede.

2. Utilização de ferramentas para automação de instalação de aplicações

Os negócios da atualidade necessitam de alta velocidade das operações para fornecer os melhores produtos com maior agilidade para o mercado. Entretanto, os ciclos de lançamento são passíveis de erro, sobre tudo aos elementos de TI invisíveis, que fogem ao controle dos departamentos de TI e não possuem aprovação da organização, o que torna mais difícil o acompanhamento das demandas do mercado de maneira geral. Uma plataforma bem estruturada para automatizar a infraestrutura e processos de implantação, estabelecendo e reforçando as boas práticas e aproveitando as habilidades adequadas, oferece suporte à TI e ajuda a superar os desafios dos negócios da atualidade.

2.1. Ansible

A plataforma de automação TI *Ansible* é de fácil utilização e transforma tarefas repetitivas e ineficientes dos ciclos de lançamento de software em processos simples, escaláveis e previsíveis.

O *Ansible* automatiza a implantação de aplicativos, a orquestração de serviços, o gerenciamento de configuração e a automação do provisionamento em nuvem permitindo que os desenvolvedores disponham de mais tempo no trabalho, auxiliando as atividades e operações a fornecer suporte às linhas de trabalho da implantação com maior facilidade. Com esses recursos unidos cria-se uma abordagem dinâmica e ágil abrangendo e coordenando a agregação de valores ao negócio.

Na implementação proposta para esta etapa, foram criados *templates* e configurações para automatizar a implantação da aplicação phpIPAM utilizando o *ansible* como agente de automação.

2.2. Criando Playbooks

Os scripts padronizados de instalação foram feitos seguindo os conceitos de *playbook* do *ansible*, que são uma série de instruções dadas pelo *ansible* para os servidores da aplicação. Estes *playbooks* são responsáveis por grande parte da automação de processos do *ansible*. A seguir serão exibidos uma sequência de ações para a configuração dos servidores e componentes do *ansible*.

```
# CRIANDO DIRETÓRIO DE TRABALHO
~$ mkdir ~/APP-phpIPAM
~$ sudo ln /etc/ansible/hosts ~/APP-phpIPAM/hosts
~$ cd APP-phpIPAM/
~/APP-phpIPAM$ ansible-galaxy init mysql -vvvv
Using /etc/ansible/ansible.cfg as config file
Opened /home/danfmsouza/.ansible_galaxy
Initial connection to galaxy_server: https://galaxy.ansible.com
Base API: https://galaxy.ansible.com/api/v1
https://galaxy.ansible.com/api/v1/platforms/?page_size
- mysql was created successfully
~/APP-phpIPAM$ ansible-galaxy init apache -vvvv
Using /etc/ansible/ansible.cfg as config file
Opened /home/danfmsouza/.ansible_galaxy
Initial connection to galaxy_server: https://galaxy.ansible.com
Base API: https://galaxy.ansible.com/api/v1
https://galaxy.ansible.com/api/v1/platforms/?page_size
- apache was created successfully
```

Quadro 3. Criando Diretório de Trabalho com Ansible-Galaxy

Antes que começar a escrever os *playbooks* de automação dos servidores, é necessário definir um inventário dos servidores gerenciados com o *ansible*. Os servidores foram configurados com autenticação por chave pública no usuário *infnet* e nas portas 10591 para o servidor nomeado MySQL e 10592 para o servidor da aplicação nomeado Apache.

```
# Inventário Padrão ~/APP-phpIPAM/hosts
[MySQL]
192.168.122.2:10591

[Apache]
192.168.122.3:10592
```

Quadro 4. Configurando Inventário

Para a configuração do playbook do servidor Apache foi escrito o seguinte script(`~/APP-phpIPAM/apache/defaults/main.yml`):

```
# Playbook Ansible automatizando servidor Apache

- hosts: Apache
  user: infnet
  become: yes
  vars:
    mysqladdr: '192.168.122.2'
    dbhost: "'localhost'"
    dbpass: "'phpipamadmin'"
    dbnewpass: "'phpipam2017'"
    siteconf: "/etc/apache2/sites-available/000-default.conf"
    phpipamdir: "/var/www/html/phpipam"

  tasks:
    - name: Atualizando o cache de repositórios APT
      action: apt update_cache=yes

    - name: Atualização do sistema
      action: apt upgrade=full

    - name: Instala o servidor apache e as dependências do phpIPAM
      apt: name={{ item }} state=present
      with_items:
        - apache2
        - mysql-client
        - php
        - php-gmp
        - php-pear
        - php-mysql
        - php-ldap
        - php-mbstring
        - libapache2-mod-php

# Instalação phpIPAM

- name: Clonando arquivos do phpIPAM
  git: repo='https://github.com/phpipam/phpipam.git' dest={{ phpipamdir }} accept_hostkey=yes update=no
  ignore_errors: true

- name: Criando arquivo de configuração config.php
  shell: "sed -e 's/{{ dbhost }}/{{ mysqladdr }}/g' {{ phpipamdir }}/config.dist.php |sed -e 's/{{ dbpass }}/{{ dbnewpass }}/g'
>{{ phpipamdir }}/config.php"

- name: Exportando Schema do Banco de dados phpipam para servidor MySQL
  mysql_db: name=phpipam state=import target={{ phpipamdir }}/db/SCHEMA.sql login_host={{ mysqladdr }} login_user=phpipam
  login_password=phpipam2017

- name: Configurando Apache DocumentRoot
  lineinfile: dest={{ siteconf }} regexp='^DocumentRoot' line="DocumentRoot /var/www/html/phpipam"
  notify:
    - restart apache2

- name: Reiniciando Apache
  service: name=apache2 state=restarted
```

Quadro 5. Playbook YAML phpIPAM `~/APP-phpIPAM/apache/defaults/main.yml`

Já para a configuração do playbook do servidor Apache foi escrito o seguinte script(`~/APP-phpIPAM/mysql/defaults/main.yml`):

```
# Playbook de Configuração do servidor MySQL

- hosts: MySQL
  user: infnet
  become: yes
  tasks:
    - name: Atualizando o cache de repositórios APT
      action: apt update_cache=yes
    - name: Atualização do sistema
      action: apt upgrade=full

    - name: Instala o servidor MySQL
      apt: name={{ item }} state=present
      with_items:
        - mysql-server
        - python-software-properties
        - python-mysqldb
    - name: Criando Banco de Dados phpipam
      mysql_db: name=phpipam state=present login_user=root login_password=MySQL211!
    - name: Criando usuário phpipam
      mysql_user: name=phpipam password=phpipam2017 priv='*.*:ALL,GRANT' state=present login_user=root login_password=MySQL211! host=192.168.122.3
```

Quadro 6. Playbook YAML MySQL `~/APP-phpIPAM/mysql/defaults/main.yml`

2.3. Executando as instruções Ansible

Após a criação do inventário e os *scripts* de instrução *ansible*, foram executados os *playbooks* com o comando *ansible-playbook*: Primeiramente foi executado o *playbook* do MySQL em `mysql/defaults/main.yml`

```
~$ ansible-playbook APP-phpIPAM/mysql/defaults/main.yml

PLAY [MySQL] *****
TASK [setup] *****
ok: [192.168.122.2]
TASK [Atualizando o cache de repositórios APT] *****
changed: [192.168.122.2]
TASK [Atualização do sistema] *****
ok: [192.168.122.2]
TASK [Instala o servidor MySQL] *****
ok: [192.168.122.2] => (item=[u'mysql-server', u'python-software-properties', u'python-mysqldb'])
TASK [Criando banco de dados phpipam] *****
changed: [192.168.122.2]
TASK [Criando usuário phpipam] *****
ok: [192.168.122.2]
PLAY RECAP *****
192.168.122.2      : ok=6    changed=2    unreachable=0    failed=0
```

Quadro 7. Saída da execução do Playbook `mysql/defaults/main.yml`

A saída da execução do *playbook* demonstra que as instruções foram executadas com sucesso para o servidor MySQL. O quadro a seguir demonstra a saída da execução do *playbook* do servidor Apache:

```
~$ ansible-playbook APP-phpIPAM/apache/defaults/main.yml
PLAY [Apache] *****
TASK [setup] *****
ok: [192.168.122.3]
TASK [Atualizando o cache de repositórios APT] *****
changed: [192.168.122.3]
TASK [Atualização do sistema] *****
ok: [192.168.122.3]
TASK [Instala o servidor apache e as dependências do phpIPAM] *****
ok: [192.168.122.3] => (item=[u'apache2', u'mysql-client', u'php', u'php-gmp', u'php-pear', u'php-mysql', u'php-ldap', u'php-strings', u'libapache2-mod-php'])
TASK [Clonando arquivos do phpIPAM] *****
ok: [192.168.122.3]
TASK [Criando arquivo de configuração config.php] *****
changed: [192.168.122.3]
[WARNING]: Consider using template or lineinfile module rather than running sed
TASK [Configurando Apache DocumentRoot] *****
ok: [192.168.122.3]
TASK [Reiniciando Apache] *****
changed: [192.168.122.3]
TASK [Exportando Schema do Banco de dados phpipam para servidor MySQL] *****
changed: [192.168.122.3]
PLAY RECAP *****
192.168.122.3 : ok=9 changed=4 unreachable=0 failed=0
```

Quadro 8. Saída da execução do Playbook apache/defaults/main.yml

A última instrução do *playbook* escrito para o servidor Apache, faz a exportação do *SCHEMA* do banco de dados padrão da aplicação *phpIPAM*. Isso foi feito devido ao fato de que, após a configuração do arquivo *config.php*, a instalação do servidor é finalizada através da interface WEB onde são definidas as informações acerca do banco de dados. Em uma instalação em larga escala não seria interessante ter que acessar cada servidor via web para finalizar a configuração. Por isso, foi incrementada esta última linha que agiliza a exportação da base de dados do *phpIPAM*.

2.4. Aplicação em Funcionamento

Com os *playbooks* escritos e executados com sucesso, a aplicação encontra-se operacional como é possível ver na imagem abaixo.

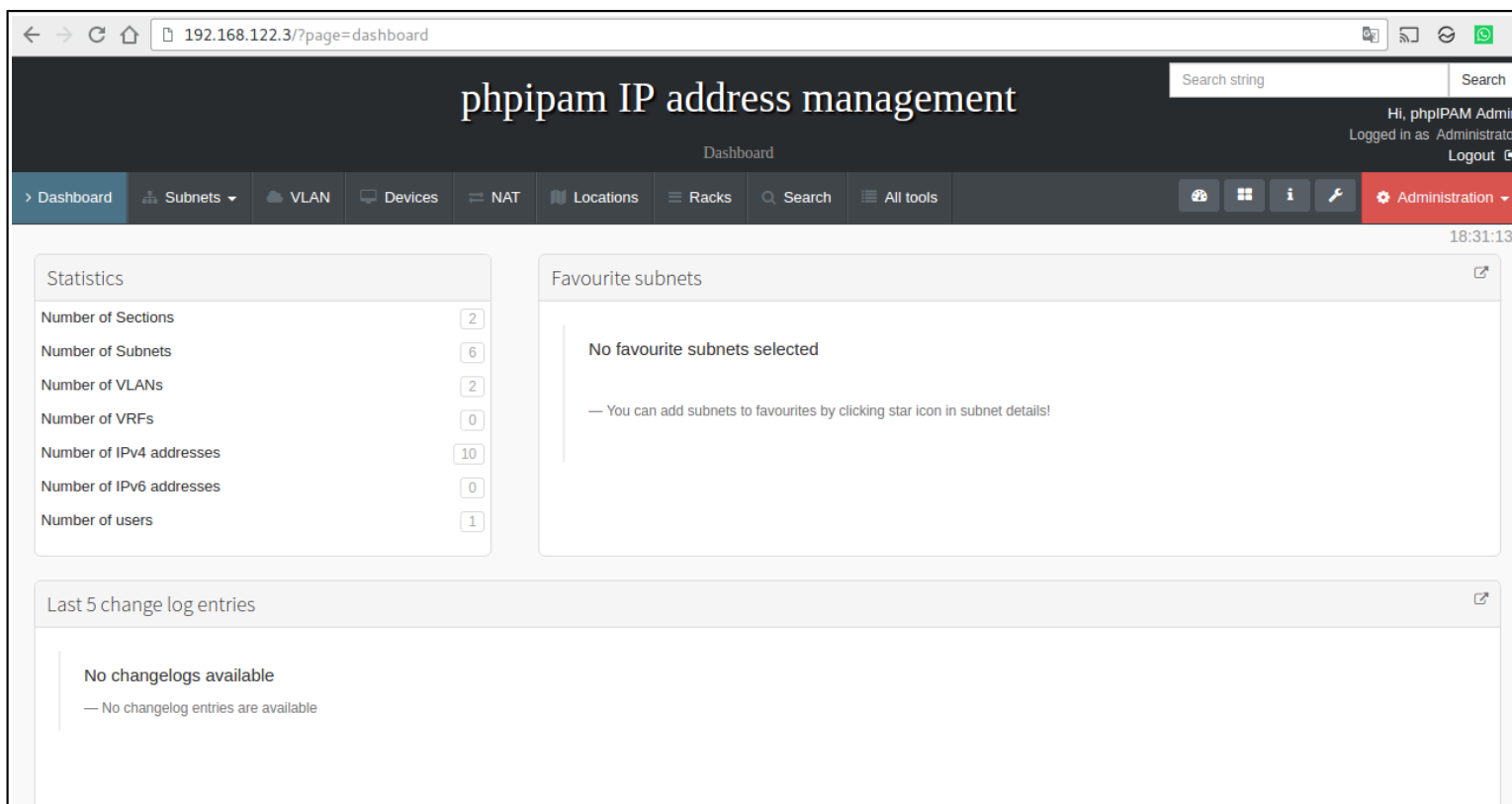


Figura 22. Aplicação acessada com as credenciais padrão

Algumas melhorias e customizações ainda podem ser realizadas para otimizar o sistema, mas a aplicação já pode ser operada perfeitamente com os *playbooks* descritos neste documento.

3. Documentar um projeto de infraestrutura para aplicações

Documentar qualquer tipo de infraestrutura é encarado diversas vezes como uma atividade chata e burocrática. No entanto, a documentação é uma boa prática e contribui com a otimização da gestão de infraestrutura gerando maior ganho na produtividade dos profissionais daquela infraestrutura.

Elaborar a documentação e atualizá-la de forma automática e iterativa é o ideal para agilizar a gestão de configuração e da própria infraestrutura em si. Uma ferramenta que torne isso possível é uma grande aliada na manutenção dos arquivos, *scripts* e documentação do que é realizado na infraestrutura de uma empresa.

O Git é uma ferramenta perfeita para solucionar esta questão. Com o Git é possível, não apenas armazenar os arquivos e documentos de uma aplicação, mas também controlar a versão de cada documento e disponibilizar para outros colaboradores o acesso às informações, para os mesmos possam trabalhar em versões paralelas à oficial. Com isso qualquer melhoria ou atualizações podem ser analisadas pelo administrador do conteúdo original que pode, até mesmo, adicionar essas mudanças no arquivo original.


A agilidade e flexibilidade dada pelo Git é muito bem-vinda quando diversos códigos estão sendo escritos para aplicações em geral, por isso será a ferramenta utilizada para documentar os *playbooks* e *scripts* da aplicação *phpIPAM*.

3.1. Criando Repositório Git

Não é o intuito deste documento demonstrar o processo de instalação do Git, mas considerando que o *desktop* utilizado usa o sistema operacional *Ubuntu*, a ferramenta foi instalada com o gerenciador de pacotes *apt*(*apt-get install git git-core*). Além disso, será considerado que já existe uma conta criada no site *github.com* que será utilizada na criação do repositório e que a chave pública ssh também já foi exportada para o *github.com*. A imagem a seguir(Imagem 4) mostra a criação do repositório diretamente no site *github.com* e o quadro seguinte(Quadro 7.) documenta os passos utilizados na inicialização do repositório Git para os playbooks utilizados na aplicação phpIPAM.

Create a new repository

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

Owner
 danfmsouza ▾


Repository name


APP-phpIPAM ✓

Great repository names are short and memorable. Need inspiration? How about [refactored-disco](#).

Description (optional)


Playbooks da aplicação distribuída phpIPAM

☒  **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: **None** ▾

Add a license: **None** ▾ 

Create repository

Figura 23. Criando repositório no Github

```
# CONFIGURANDO INFORMAÇÕES GLOBAIS
~$ git config --global user.name "Daniel Souza"
~$ git config --global user.email "kaedasm@gmail.com"
~$ cd APP-phpIPAM/

# DEFININDO EDITOR PADRÃO DO GIT
~/APP-phpIPAM$ git config --global core.editor vim

# CUSTOMIZANDO GIT PARA MELHOR VISUALIZAÇÃO (Git Colorido)
~/APP-phpIPAM$ git config --global core.editor vim
~/APP-phpIPAM$ git config --global color.branch auto
~/APP-phpIPAM$ git config --global color.diff auto
~/APP-phpIPAM$ git config --global color.grep auto
~/APP-phpIPAM$ git config --global color.status auto

# INICIALIZANDO E COMMISSIONANDO UM REPOSITÓRIO DO DIRETÓRIO APP-phpIPAM
~/APP-phpIPAM$ echo "# APP-phpIPAM" >> README.md
~/APP-phpIPAM$ git init

Initialized empty Git repository in /home/danfmsouza/APP-phpIPAM/.git/
~/APP-phpIPAM$ git add README.md
~/APP-phpIPAM$ git commit -m "Primeiro Comissionamento"

[master a2585d6] Primeiro Comissionamento
 1 file changed, 1 insertion(+)

~/APP-phpIPAM$ git remote add origin git@github.com:danfmsouza/APP-phpIPAM.git
~/APP-phpIPAM$ git push -u origin master

Counting objects: 41, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (22/22), done.
Writing objects: 100% (41/41), 6.41 KiB | 0 bytes/s, done.
Total 41 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To git@github.com:danfmsouza/APP-phpIPAM.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

Quadro 9. Criando Repositório no Github

3.2. Realizando o *Push* das Alterações no Repositório

Uma das funções mais importantes do Git é a capacidade de enviar alterações realizadas nos códigos. Para demonstrar o processo de *commit* e *push* do Git na atualização do repositório, o arquivo README.md será editado e um novo comissionamento será feito mostrando a atualização realizada.

```
# ALTERANDO ARQUIVO README.md

~/APP-phpIPAM$ echo "Repositório de Documentação da Aplicação phpIPAM para os servidores da Giro Telecom.
Para Executar uma nova instalação do phpIPAM utilize os main Playbooks em defaults de ambos os diretórios (mysql e apache)." >>
README.md

# VERIFICANDO STATUS DO REPOSITÓRIO

~/APP-phpIPAM$ git status

No ramo master

Your branch is up-to-date with 'origin/master'.

Mudanças a serem submetidas:
  (use "git reset HEAD <file>..." to unstage)

Changes not staged for commit:
  (utilize "git add <arquivo>..." para atualizar o que será submetido)
  (utilize "git checkout -- <arquivo>..." para descartar mudanças no diretório de trabalho)
    modified:   README.md

# REALIZANDO NOVO COMISSIONAMENTO E ATUALIZAÇÃO NO GITHUB

~/APP-phpIPAM$ git add *

~/APP-phpIPAM$ git commit -m "Segundo Comissionamento, Criando Descrição no arquivo README.md"


[master 141dac6] Segundo Comissionamento, Criando Descrição no arquivo README.md
 1 files changed, 5 insertions(+)





~/APP-phpIPAM$ git push -u origin master


Warning: Permanently added the RSA host key for IP address '192.30.253.113' to the list of known hosts.
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 622 bytes | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To git@github.com:danfmsouza/APP-phpIPAM.git
 a2585d6..141dac6 master -> master
Branch master set up to track remote branch master from origin.
```

Quadro 10. Demonstrando processo de alterações no repositório Git

Com os passos descritos acima, é possível ver que as alterações foram realizadas com sucesso. Para confirmar se o repositório foi corretamente alterado, basta visualizar o link para o repositório no site *github.com* com o link de acesso público <https://github.com/danfmsouza/APP-phpIPAM>(figura 24.).


danfmsouza Segundo Comissionamento, Criando Descrição no arquivo README.md
 Latest commit 141dac6 6 minutes ago

 apache	Primeiro Comissionamento dos Arquivos	an hour ago
 mysql	Primeiro Comissionamento dos Arquivos	an hour ago
 README.md	Segundo Comissionamento, Criando Descrição no arquivo README.md	6 minutes ago
 hosts	Segundo Comissionamento, Criando Descrição no arquivo README.md	6 minutes ago

 **README.md**

APP-phpIPAM

APP-phpIPAM

Repositório de Documentação da Aplicação phpIPAM para os servidores da Giro Telecom. Para Executar uma nova instalação do phpIPAM utilize os main Playbooks em defaults de ambos os diretórios (mysql e apache).

Figura 24. Repositório Atualizado no github.com

Com o link de acesso ao github(<https://github.com/danfmsouza/APP-phpIPAM>) em estado público, qualquer pessoal terá acesso aos arquivos e poderão realizar *forks* e ver qualquer alteração que seja realizada. É possível tornar o repositório particular, mas por motivos demonstrativos o repositório foi criado como público. Apesar de qualquer um ter acesso às informações contidas no repositório, apenas o usuário dono dele poderá realizar *pushs* e *commits* nele.

4. Selecionando a plataforma de execução para uma aplicação

O mundo da tecnologia da informação, já a algum tempo, vem sido invadido pela virtualização, sobre tudo da computação em nuvem. Este cenário é irreversível hoje, dadas as vantagens de um sistema virtualizado e integrado na nuvem. Aplicações que utilizam este conceito se beneficiam com a segurança da estabilidade e confiabilidade propostos. A facilidade de ativar um serviço, ou mesmo, recuperar-se de falhas ou até mesmo escalar dinamicamente o uso de uma aplicação de acordo com sua demanda, são vantagens chave para qualquer negócio.

Com este cenário em mente, nesta etapa será demonstrado na prática a instalação da aplicação phpIPAM utilizando os conceitos de *containers* abordados pelo Docker. Diferente de um ambiente de virtualização comum, onde existe um sistema operacional completo e isolado, no Docker há recursos isolados que compartilham bibliotecas de *kernel* em comum, isto é, entre o servidor *host* e o *container*, que só é possível pois o Docker utiliza como backend o LXC(LinuX Containers). Uma arquitetura em contêiner para executa uma aplicação com os recursos do *host* de forma mais econômica.

```
# ~/.$ sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
** SAÍDA DO COMANDO OMITIDA **
# ~/.$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
** SAÍDA DO COMANDO OMITIDA **
# ~/.$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
** SAÍDA DO COMANDO OMITIDA **
# ~/.$ sudo apt-get update
*** SAÍDA DO COMANDO OMITIDA ***
# ~/.$ sudo apt-get install docker-ce
*** SAÍDA DO COMANDO OMITIDA ***
# ~/.$ docker run --name ipanDB -e MYSQL_ROOT_PASSWORD=phpIPAM2017 mysql:5.6 &
*** DOWNLOAD DA IMAGEM OMITIDO ***
*** LOGS DE INICIALIZAÇÃO OMITIDOS ***
# ~/.$ docker run -ti -p 80:80 --name phpIPAM --link ipanDB:mysql michaelholtttech/phpipam &
*** DOWNLOAD DA IMAGEM OMITIDO ***
*** LOGS DE INICIALIZAÇÃO OMITIDOS ***
# ~/.$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
386215fa15e7	michaelholtttech/phpipam	"apache2-foreground"	4 minutes ago	Up 3 minutes	0.0.0.0:80->80/tcp	phpIPAM
cdfab9a7ccea	mysql:5.6	"docker-entrypoint..."	29 minutes ago	Up 29 minutes	3306/tcp	ipanDB

Quadro 11. Rodando phpIPAM com docker

Com os comandos acima, a instalação do gerenciador de contêineres Docker e dos contêineres do banco de dados e da aplicação apache com phpipam(ipamDB e phpIPAM), a aplicação phpIPAM já encontra-se disponível para a sua configuração via interface web.

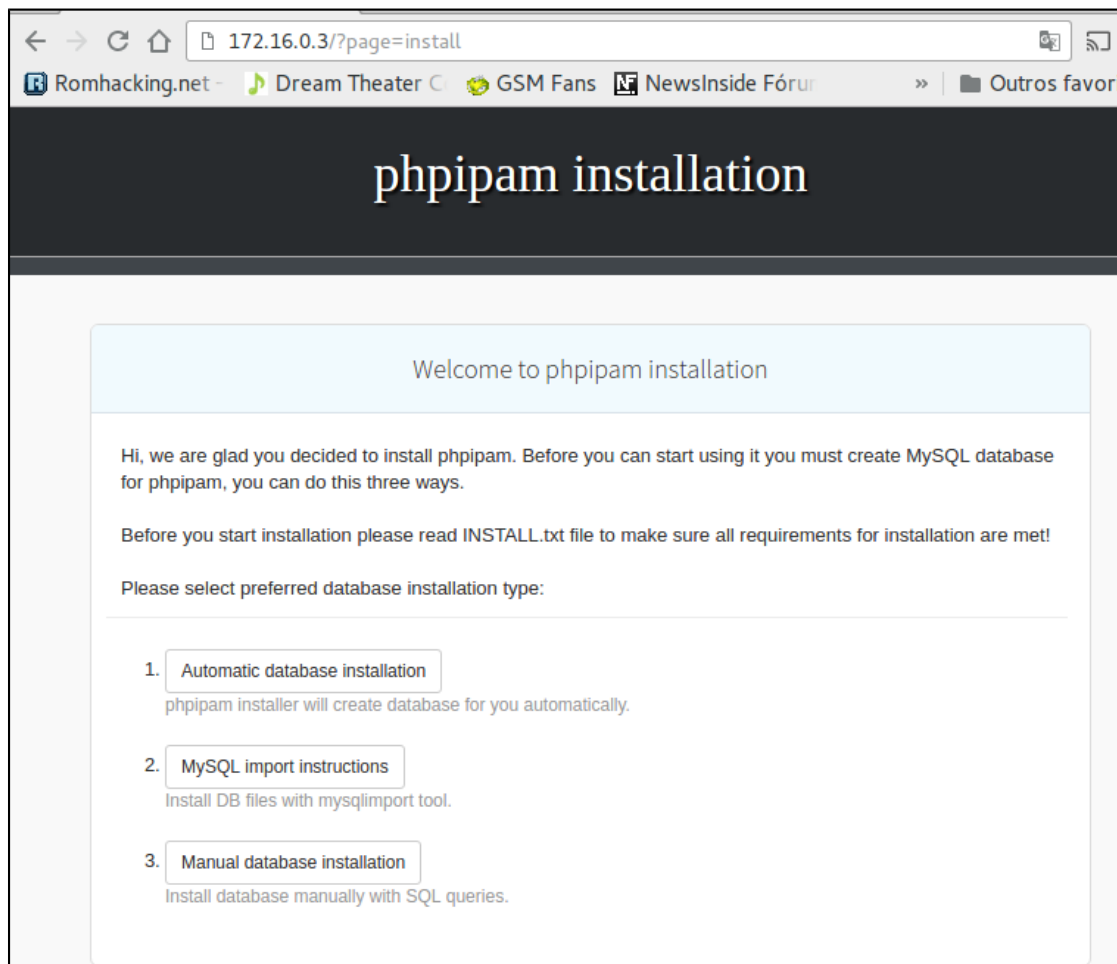


Figura 25. phpIPAM sendo executado através dos contêineres Docker

De certa forma a instalação com este método é bem simples e prática. Para um administrador do sistema é muito interessante ter esse tipo de agilidade, sobretudo em um cenário de recuperação de falhas.

Apesar de ser uma aplicação distribuída, esta aplicação está rodando em dois *containers* diferentes ainda está sendo executada no mesmo host. O docker tem alguns recursos de integração de redes para que *hosts* Docker distintos possam *linkar containers* entre si.

5. Gerenciamento de nuvem com Ansible

Considerando os conceitos básicos do gerenciamento de aplicações em nuvens, o ansible foi utilizado para executar os *containers* docker de maneira automatizada. A seguir serão mostrados o *playbook* e as roles para esta implementação

5.1. Role docker (Ativando suporte ao docker)

```
# ~/APP-phpIPAM$ ansible-galaxy init docker
# - docker was created successfully
# ~/APP-phpIPAM$ vim docker/tasks/main.yml

---
- name: Instalando Dependências do Docker
  apt:
    name={{ item }}
    update_cache=yes
  with_items:
    - python-dev
    - python-setuptools

- name: Instalando o PIP via easy_install
  easy_install:
    name=pip

- name: Instalando o módulo docker-py
  pip:
    name=docker-py
    state=present

- name: Adicionando Repositório APT para o Docker
  apt_repository:
    repo='deb https://apt.dockerproject.org/repo ubuntu-{{ ansible_distribution_release }} main'
    state=present

- name: Importando Chave do Repositório Docker Project
  apt_key:
    url=https://apt.dockerproject.org/gpg
    state=present
    id=2C52609D

- name: Instalando Pacote Docker
  apt:
    name=docker-engine
    update_cache=yes
```

Quadro 12. Ativando Suporte ao Serviço Docker

```
# ~/APP-phpIPAM$ vim mysql-docker/templates/default_docker.j2
# Docker Upstart and SysVinit configuration file.
# Use DOCKER_OPTS to modify the daemon startup options.
{% if docker_opts is defined %}
DOCKER_OPTS="{{ docker_opts | join(' ') }}"
{% endif %}
```

Quadro 13. Criando Template de Configuração Docker

Nos quadros acima primeiro é realizada a criação da role *docker* que se encarregará de instalar o docker e suas dependências, incluindo o repositório APT. Já no quadro seguinte, é criado um modelo de configuração docker.

O role docker deverá ser executado antes dos demais pois as dependências da aplicação também se aplicam ao próprio ansible para a execução do módulo *docker* na automação dos processos.

5.2. Role mysql-docker (Rodando o Contêiner MySQL)

Com o Docker devidamente instalado na plataforma, o primeiro passo para a ativação da aplicação é a instalação do servidor de banco de dados MySQL. Essa ativação será realizada utilizando um contêiner MySQL suportado pela comunidade disponível em <http://hub.docker.com>.

```
# ~/APP-phpIPAM$ ansible-galaxy init mysql-docker
# - mysql-docker was created successfully
# ~/APP-phpIPAM$ vim mysql-docker/tasks/main.yml
---
- name: "Executa o container MySQL"
  docker_container:
    name: ipanDB
    image: mysql:5.6
    env:
      MYSQL_ROOT_PASSWORD: phpIPAM2017
```

Quadro 14. Ativando Contêiner MySQL

5.3. Role phpipam-docker (Rodando o Contêiner phpIPAM)

A role phpipam-docker executa o container a partir de uma imagem mantida no repositório da comunidade docker. No quadro abaixo é executado o ansible-galaxy para a criação da role e em seguida é utilizado o editor de textos/IDE *vim* para escrever o script ansible adequado para esta operação.

```
# :~/APP-phpIPAM$ ansible-galaxy init phpipam-docker
# - phpipam-docker was created successfully
# :~/APP-phpIPAM$ vim phpipam-docker/tasks/main.yml
```

```
—
- name: Executa o container phpIPAM
  docker_container:
    name: phpIPAM
    image: "michaelholttech/phpipam"
    links:
      - "ipamDB:mysql"
    ports:
      - 80:80
```

Quadro 15. Ativando Contêiner phpIPAM

5.4. Playbook ansible docker-playbook.yml

Por fim e não menos importante, é criado o *playbook* e inventário que executará as roles previamente escritas para a execução da aplicação.

```
# :~/APP-phpIPAM$ vim docker-playbook.yml
```

```
—
- name: Instalando Docker e suas dependências
  hosts: docker
  remote_user: noc
  become: yes
  roles:
    - docker
```

```
- name: Rodando Container ipamDB
  hosts: docker
  remote_user: noc
  become: yes
  roles:
    - mysql-docker
```

```
- name: Rodando Container phpIPAM
  hosts: docker
  remote_user: noc
  become: yes
  roles:
    - phpipam-docker
```

```
# :~/APP-phpIPAM$ echo "
[docker]
186.216.192.82
" >>hosts
```

Quadro 16. Criando Playbook e Configurando o Inventário

Com a criação do Playbook e atualização do inventário, agora basta executar o *playbook* com o para iniciar a implementação dos contêineres que executarão o phpIPAM no servidor.

```

~/APP-phpIPAM$ ansible-playbook -i hosts docker-playbook.yml
PLAY [Instalando Docker e suas dependências] *****
TASK [setup] *****
ok: [186.216.192.82]
TASK [docker : Instalando Dependências do Docker] *****
ok: [186.216.192.82] => (item=[u'python-dev', u'python-setuptools', u'python-software-properties'])
TASK [docker : Instalando o PIP via easy_install] *****
ok: [186.216.192.82]
TASK [docker : Instalando o módulo docker-py] *****
ok: [186.216.192.82]
TASK [docker : Adicionando Repositório APT para o Docker] *****
ok: [186.216.192.82]
TASK [docker : Importando Chave do Repositório Docker Project] *****
ok: [186.216.192.82]
TASK [docker : Instalando Pacote Docker] *****
ok: [186.216.192.82]
TASK [docker : Criando grupo de Usuários Docker] *****
ok: [186.216.192.82]
TASK [docker : Adicionando usuários ao grupo Docker] *****
skipping: [186.216.192.82] => (item=noc)
TASK [docker : Configurando o Docker] *****
ok: [186.216.192.82]
PLAY [Rodando Container ipamDB] *****
TASK [setup] *****
ok: [186.216.192.82]
TASK [mysql-docker : Executa o container MySQL] *****
changed: [186.216.192.82]
PLAY [Rodando Container phpIPAM] *****
TASK [setup] *****
ok: [186.216.192.82]
TASK [phpipam-docker : Executa o container phpIPAM] *****
changed: [186.216.192.82]
PLAY RECAP *****
186.216.192.82      : ok=13  changed=2    unreachable=0    failed=0

```

Quadro 17. Executando o Playbook

Com a execução bem-sucedida da aplicação através do ansible com o docker, já é possível acessar a aplicação phpIPAM pelo IP definido no inventário.

O quadro a seguir mostra a janela da aplicação phpIPAM rodando através dos *containers*.

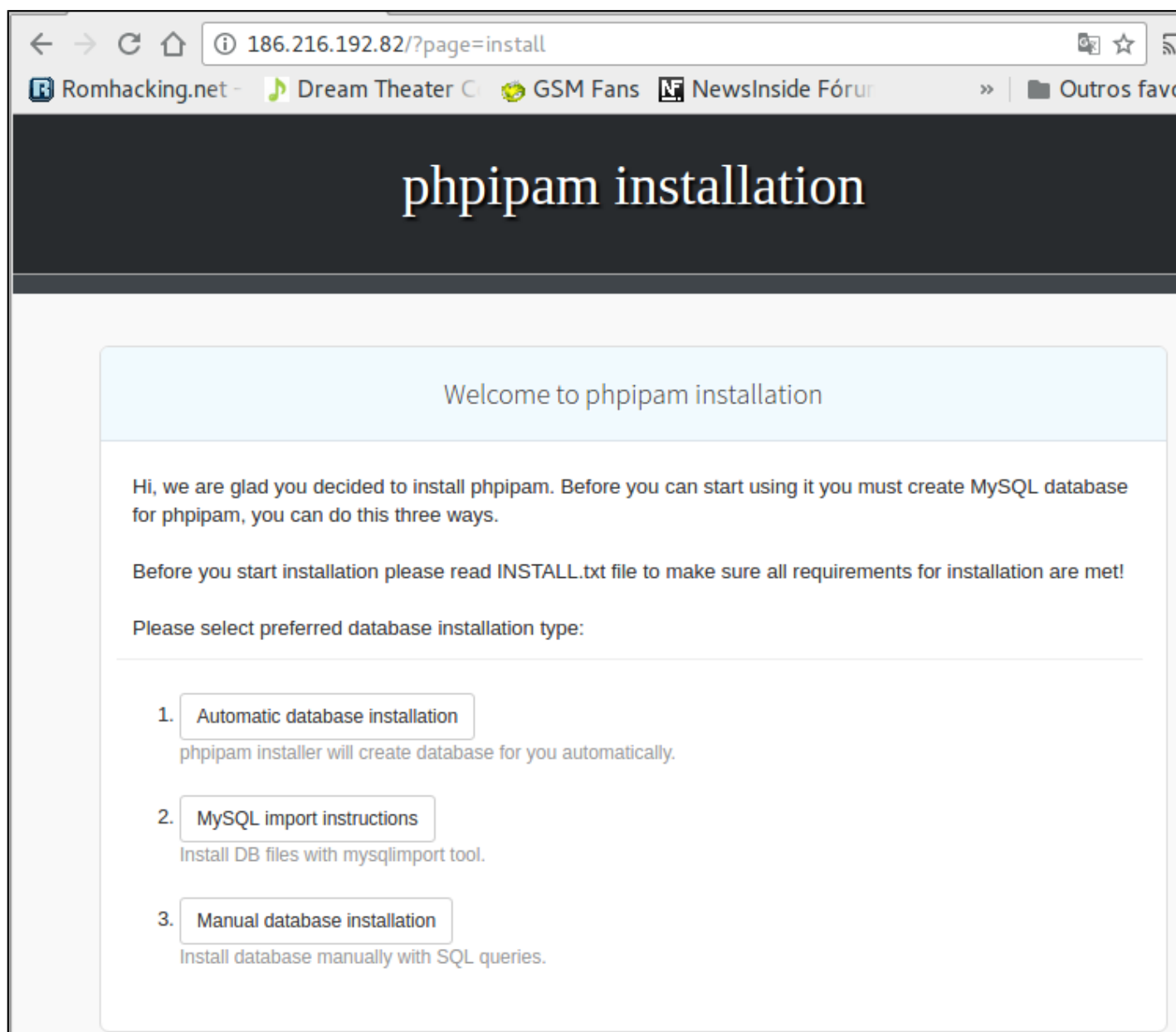


Figura 26. Cliente acessando a aplicação implementada com Docker e Ansible

Com o *deploy* da aplicação realizado com sucesso, graças a automação de processos realizadas com o ansible, é possível acessar a aplicação normalmente através do endereço IP do host onde os contêineres foram instalados.

É possível, com o docker, realizar um acesso direto à aplicação sem utilizar o acesso redirecionado criado pelo host. Para isso seria necessário gerenciar os métodos de conexão de rede e hubs virtuais criados pela aplicação.

5.5. Atualizando a Documentação com o GIT

Como os roles e o playbook foram criados dentro do repositório anteriormente criado para o modelo tradicional de implementação, o processo de documentação através da ferramenta GIT se torna bem simples. Alguns poucos comandos definem quais arquivos serão atualizados no repositório.

```
# ~/APP-phpIPAM$ git add hosts docker-playbook.yml docker mysql-docker phpipam-docker
# ~/APP-phpIPAM$ git commit -m "Comissionando Arquivos do TP5"

[master 75f2d2c] Comissionando Arquivos do TP5
 32 files changed, 1017 insertions(+)
*** TRECHO DA SAÍDA OMITIDO ***

# ~/APP-phpIPAM$ git push -u origin master

Counting objects: 44, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (21/21), done.
Writing objects: 100% (44/44), 4.29 KiB | 0 bytes/s, done.
Total 44 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To git@github.com:danfmsouza/APP-phpIPAM.git
   ba291bc..3ddcd51  master -> master
Branch master set up to track remote branch master from origin.
```

Quadro 18. Atualizando Repositório GIT

Com isso todas os arquivos alterados para a operação foram atualizados no repositório git <https://github.com/danfmsouza/APP-phpIPAM> e podem ser acessados a qualquer momento.

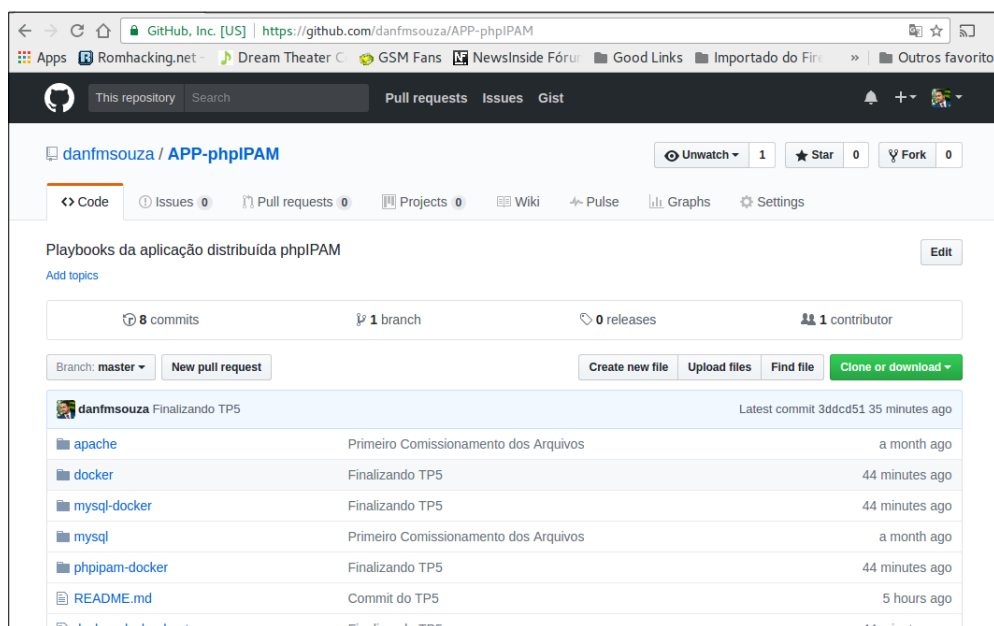


Figura 27. Repositório GIT APP-phpIPAM com os novos Roles e Playbooks

Conclusão

Segundo os prazos apresentados no ***Cronograma de Implantação***, o projeto pode ser executado dentro dos prazos indicados com uma certa folga em alguns momentos. O Ansible foi uma ferramenta que otimizou muito o processo de implementação da aplicação, o que economizou bastante tempo em alguns pontos.

Os recursos planejados para a implementação atenderam bem a proposta, além disso, a plataforma de virtualização dispõe de ferramentas que auxiliam no escalonamento das máquinas virtuais, dando assim maior vida para a aplicação no ambiente instalado.

Todas as funcionalidades da aplicação, tanto da plataforma quanto da aplicação funcionaram como o esperado. A implementação de ambas não sofreu dificuldade e a interpretação distribuída da aplicação respondeu de acordo com o planejado.

Futuramente a aplicação poderá ser expandida de acordo com as necessidades, uma vez que a plataforma onde foi implementada permite o escalonamento de recursos e alta disponibilidade. Outro ponto interessante é que da forma em que a aplicação foi concebida e planejada, poderá ser migrada para qualquer outra das soluções apresentadas sem muito esforço, e no pior dos casos poderá ser reimplementada e restaurada a partir de um backup previamente criado.

A proposta de soluções de arquitetura de infraestrutura para aplicações de acordo com o cenário dentro do contexto deste documento, é apenas um exemplo de como o preparo antes da implementação de um recurso de software. Sistemas executando aplicações podem ser muito complexos, sobretudo quando há muitos elementos e servidores para serem gerenciados. A integração do desenvolvimento com a operação pode ser uma ótima maneira de otimizar a forma em que as aplicações são executadas na infraestrutura. Utilizando ferramentas e modelos como o SDDC, toda a complexidade da infraestrutura do data center fica facilmente gerenciada como um serviço único,

o que agiliza e dinamiza os seus processos, deixando tempo e espaço para que melhorias de usabilidade e performance possam surgir com maior frequência, além de trazer as soluções de problemas e bugs muito mais rapidamente.

Com o pouco apresentado sobre o *ansible* até agora, é perceptível que esta é uma poderosa e extremamente útil na gestão da configuração e orquestração dos servidores de uma infraestrutura. No exemplo apresentado, foram executadas tarefas em dois *hosts* de uma aplicação distribuída, mas é compreensível que podemos executar em centenas ou milhares seguindo o mesmo conceito, comprovando o poder de orquestração do *ansible*.

Manter um repositório simples com Git é uma ótima maneira de ter um controle pró-ativo das mudanças nos códigos de uma aplicação ou dos *playbooks*, como no caso deste documento. O Git é um DVCS muito poderoso e conhecer bem os seus recursos certamente ajudará a simplificar e melhorar o fluxo de trabalho no VCS. Fora o fato de poder ter um repositório bem documentado dos códigos e documentos de uma determinada infraestrutura.

O cenário proposto permite que modelos distintos de arquitetura sejam implementados. Modelos de arquitetura baseados em contêineres são uma grande realidade no mercado, sobretudo no que tange a computação e nuvem. Ser capaz de implementar e gerenciar esta tecnologia é fundamental para o sucesso de um negócio baseado em serviços web, por exemplo, como é o caso do phpIPAM.

Unindo os conceitos de orquestração do *ansible* com a tecnologia de desenvolvimento em ambiente em nuvem do *docker*, aplicações com implementação complexa e alto nível de demanda de suporte podem ser rapidamente implementadas e mantidas por qualquer desenvolvedor. Unir estes recursos é fundamental para o rápido crescimento de qualquer negócio, que baseie sua tecnologia da informação em componentes virtualizados para a aplicação em nuvem.

Referências e Bibliografias

FUNG, Han Ping. A Glimpse into Software Defined Data Center. Revista de Administração de Roraima-RARR, Boa Vista, vol. 4, jul.-dez 2014

BRANDT, T., Ye, T., Markus, H., & Dirk, N. (2012). AUTONOMIC MANAGEMENT OF SOFTWARE AS A SERVICE SYSTEMS WITH MULTIPLE QUALITY OF SERVICE CLASSES. ECIS2012. Barcelona.

CITRIX. (08 de 2013). Design Your Cloud Strategy for Long-term Success. Acesso em 10 de Fevereiro de 2017, disponível em citrix.com: http://www.citrix.com/content/dam/citrix/en_us/documents/products-solutions/design-you-cloud-strategy-for-long-term-success.pdf

CLARK, T., Reddy, I., & Walton, D. (2011). Cisco on Cisco Best Practice IT as a Services Organization . San Jose, CA, EUA.

VMWARE. (31 de 8 de 2010). VMware Introduces End-User Computing Strategy and Products to Drive IT as a Service. SAN FRANCISCO.

Underdahl B & Novak R. (2014). Software Defined Data Centers For Dummies. John Wiley & Sons, Inc

CITRIX. Definição de VDI. Disponível em: 11 de fevereiro de 2017: <https://www.citrix.com/glossary/vdi.html>

Georgina S & Roland C. (2014). Delivering the Software Defined Data Center "VMWare & F5 Publications."

Red Hat, Inc 2017 – “Ansible Official Documentation” no site docs.ansible.com link <http://docs.ansible.com/ansible/index.html> acesso em 28/02/2017

Petkovsek, Miha – “phpIPAM Official Documentation” no site phpipam.net no link <https://phpipam.net/documents/all-documents/> acesso em 20/01/2017

Oracle Corporation and/or its affiliates - “Documentation” no site dev.mysql.com link <https://dev.mysql.com/doc/> acesso em 27/02/2017

Chacon, Scott and Straub, Ben - *“Pro Git: Everithing You Need To Know About GIT”* Apress; 2nd ed. 2014 edition - ISBN-13: 978-1484200773

TURNBULL, James - *“The Docker Book: Containerization is the new virtualization”* Ed 17030 (12 de julho de 2014) – ISBN: 978-0-9888202-0-3

VMWARE – vCenter Server no site *vmware.com* pelo link <http://www.vmware.com/products/vcenter-server.html> acessado em 19/03/2017

VMWARE - vCenter Server Prerequisites – Documentação no site *vmware.com* pelo link https://pubs.vmware.com/vsphere-50/index.jsp?topic=%2Fcom.vmware.vsphere.install.doc_50%2FGUID-C6AF2766-1AD0-41FD-B591-75D37DDB281F.html acessado em 19/03/2017

DEBIAN – Bind9 Configuration – no site *debian.org* pelo link <https://wiki.debian.org/Bind9> acessado em 19/03/2017