**INSTITUTO INFNET**

**ESCOLA SUPERIOR DE TECNOLOGIA DA**

**INFORMAÇÃO**

**GRADUAÇÃO EM GESTÃO DE TI**



**PROJETO DE BLOCO – ARQUITETURA DE**

**INFRAESTRUTURA DE APLICAÇÕES**

**TESTE DE PERFORMANCE DA ETAPA 4**

**ALUNO: ALAN AGUINAGA**

**E-MAIL: alansenairj@gmail.com**

**TURMA: GRLRDC01C1-M1-L1 - Classe 1 - Manhã**



RDC - GRADUAÇÃO EM REDE DE COMPUTADORES

ALAN FERREIRA AGUINAGA

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

Rio de Janeiro

2018

**Sumário**

[1. Introdução 3](#_Toc526890777)

[2. Objetivo. 3](#_Toc526890778)

[3. Justificativa 3](#_Toc526890779)

[4. Desenvolvimento 3](#_Toc526890780)

[4.1 Instalação do Docker 3](#_Toc526890781)

[4.2 Explicação da aplicação. 9](#_Toc526890782)

[5. Bibliografia 10](#_Toc526890783)

# 1. Introdução

# Faça upload do conteúdo completo de seu Projeto de Bloco até aqui, incluindo todos os capítulos apresentados nos TPs 6, 7 e 8, mais uma versão inicial do capítulo 4/Conclusões.

# Além dos requisitos apresentados para os capítulos anteriores, suas conclusões devem incluir:

# Se o prazo estabelecido no início do projeto foi adequado para a execução (não há problema se levou mais tempo que o planejado, mas se levou, explique as razões).

# Se os recursos planejados (quantidade de memória, disco, etc.) foram suficientes para colocar a solução em funcionamento. Se não foram, explique as razões e sugira uma nova configuração.

# Se as funcionalidades previstas na solução original funcionaram a contento. Se não foram, explique as razões.

# Quais seriam as melhorias futuras que poderiam ser feitas no projeto executado, adotando soluções adicionais, aperfeiçoando instalações ou tornando o cenário mais complexo.

# Antes da entrega final, todo o conteúdo de seu trabalho (mesmo que sejam apenas documentos do Word e PDFs) deve ser armazenado em um repositório GitHub, cujo link deve estar contido no corpo do trabalho.

# 2. Objetivo.

Executar o trabalho final do curso, reunindo tudo que foi passado até agora nos TPs

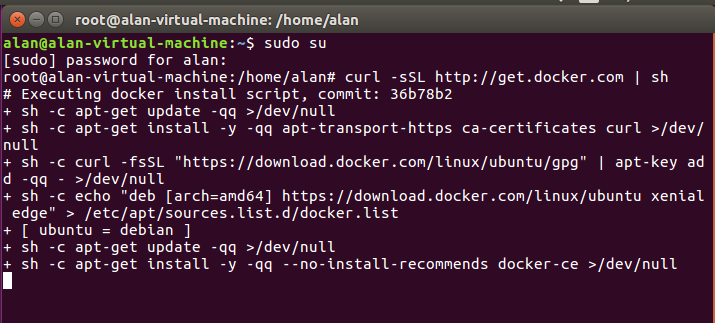
# 3. Justificativa

Demonstrar a instalação de contêineres e aplicações distribuídas para fixar os conhecimentos sobre as ferramentas aprendidas até agora.

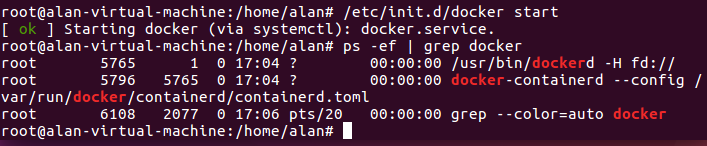
# 4. Desenvolvimento

# 4.1 Instalação do Docker

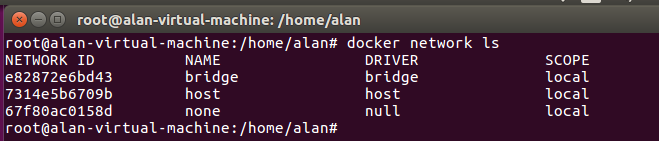
Foi instalado o Docker através de um script de instalação.



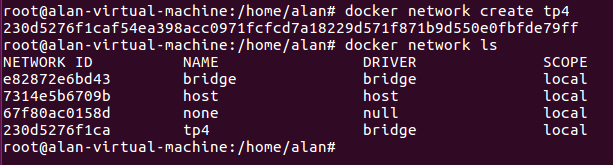
Foi realizada a inicialização para a verificação do processo no sistema.



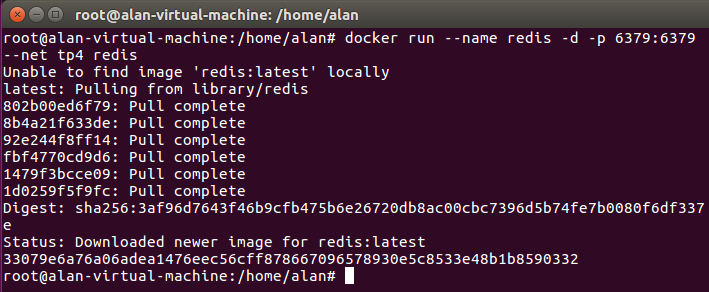
Foi criada uma rede padrão pelo Docker.



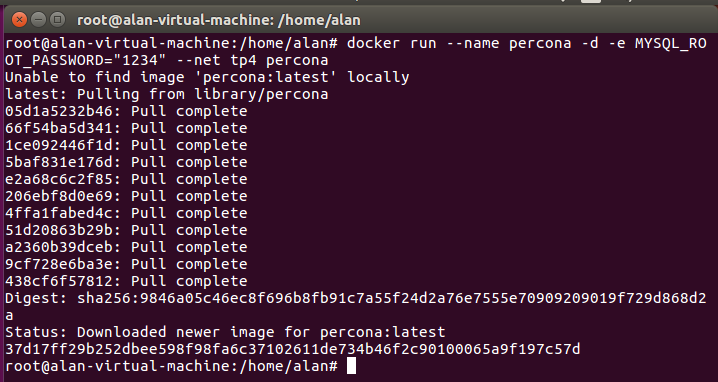
Foi criada uma rede com o nome “tp4”.



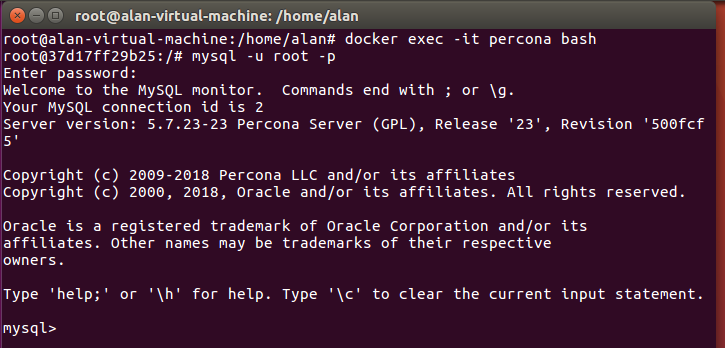
Foi instalado uma máquina Ubuntu em um container que contém o Wordpress instalado



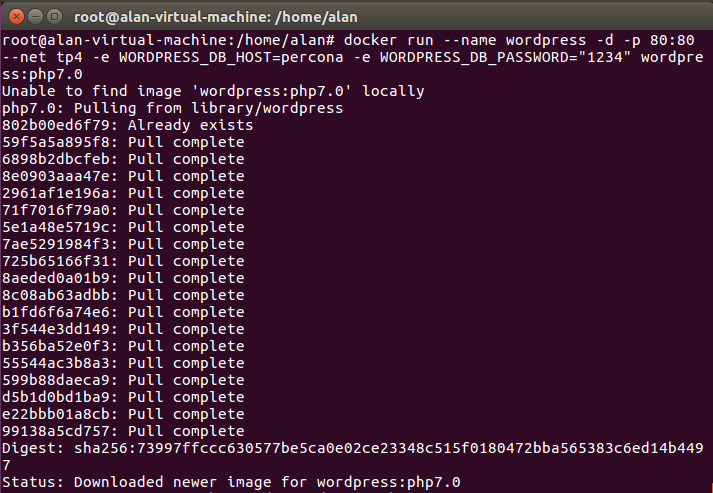
Foi instalada a aplicação com os seus respectivos bancos de dados.



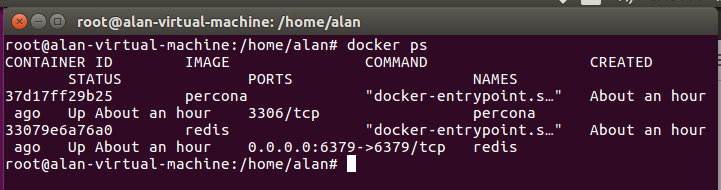
Foi executado o bash da aplicação do banco de dados para definição de senha do usuário root.



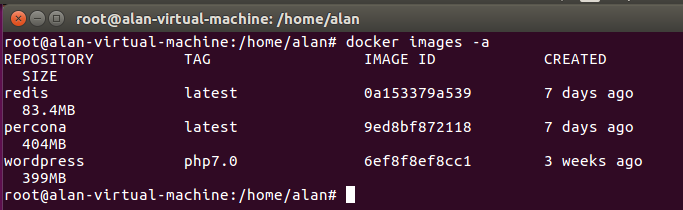
Foi inicializada a instalação da aplicação.



Com o comando abaixo, foram listados os 2 containers percona e redis.

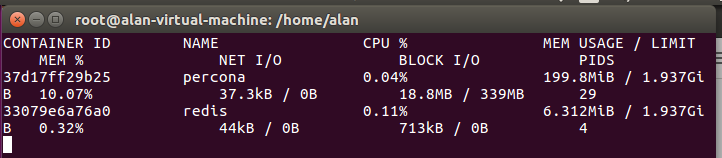


Essas são as imagens baixadas e hospedadas pelo Docker.

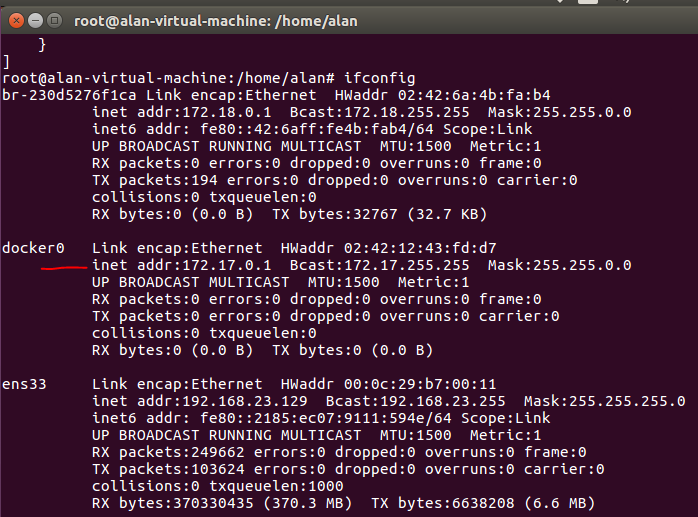


Com o comando abaixo, as estatísticas de uso dos recursos da máquina foram exibidas.

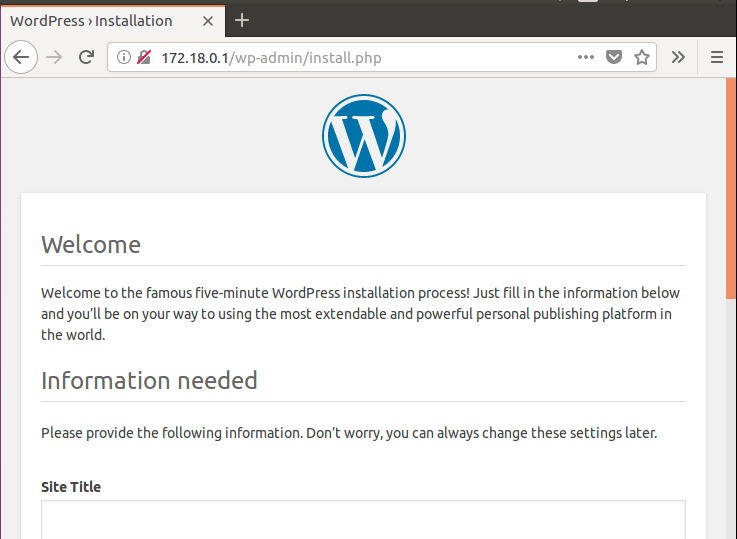




A interface de rede do Docker exibe o IP para acessar a aplicação.



A aplicação se encontra hospedada e funcional.



# 4.2 Explicação da aplicação.

Em se tratando de Docker, existe uma grande facilidade em se instalar a funcionalidade em um sistema Linux. Esse sistema tem grande suporte para a utilização de contêineres de aplicação.

Docker é uma plataforma de código aberto feito pela Google que usa uma linguagem chamada Go. O Docker necessita de um kernel Linux atualizado para operar. O Docker usa um recurso que já existe no Linux, porem conta com um site que hospeda, de maneira similar aos repositórios de aplicação, imagens de sistemas com aplicações já instaladas e configuradas.

Um contêiner pode ser até 7 vezes mais rápido que uma máquina virtual. Ele usa recursos do sistema que trata os processos (chroot) de maneira modular, permitindo o isolamento de um ambiente. Cada imagem hospedada serve como base de um sistema operacional. Os contêineres s são os locais que as aplicações são modificadas.

O Docker apresenta uma maneira muito segura para executar, pro exemplo, a implementação e teste de uma aplicação, usando uma máquina física, e disponibilizando diversos ambientes de teste para o desenvolvimento de software. É possível prover diversos contêineres diferentes para uma mesma imagem, de forma a isolar cada um deles.

O Docker utiliza uma rede paralela própria que conecta as imagens e seus respectivos contêineres em uma rede própria e paralela a rede do sistema. A classe da rede padrão do Docker é a classe B. é importante criar uma rede que não se misture com a rede física real existente, por isso o Docker cria por padrão uma interface de rede e a insere dentro de sua própria rede.

No caso da instalação e disponibilização do Wordpress através do Docker, foi escolhido a instalação de uma imagem Ubuntu Linux versão 16.04 para evitar incompatibilidades.

Um container pode ser comparado a um processo. O contêiner só fica exposto em usa utilização. Ao sair do contêiner o processo é encerrado.

No caso da hospedagem do Wordpress foram usadas 3 imagens diferentes. A imagem redis é um servidor de dicionário para armazenamento persistente, disponibilizado no repositório Docker em: <https://hub.docker.com/_/redis/>

A outra imagem percona tem a finalidade de armazenar o banco de dados da aplicação, nesse caso o banco de dados mysql. <https://hub.docker.com/_/percona/>

A ultima imagem é Wordpress que contém a aplicação que servirá de ambiente de desenvolvimento do site. Essa aplicação roda nessa imagem através do servidos apache.

Com essas 3 imagens é possível gerar um container de aplicação Wordpress operacional.

A empresa necessita criar um ambiente de desenvolvimento que permita aos alunos provisionar instâncias Linux preparadas para prover uma página gerada pelo framework Wordpress em sua rede sem fio local. O principal objetivo é prover infraestrutura como serviço para permitir o desenvolvimento de aplicativos e sites no futuro, melhorando a disponibilidade de recursos e inserindo o desenvolvimento dos projetos em um controle de versionamento do Github. Essa empresa necessita das seguintes atribuições de seu TI

* Ambiente de desktops virtuais para criação do site
* Ambiente de produção separado de ambiente de testes
* Armazenamento de seu banco de dados
* Backup e redundância
* Performance elástica das instâncias para dias de forte demanda de acesso
* Balanceamento de carga para lidar com dias de forte demanda de acesso
* Firewall e roteamento

Relevância

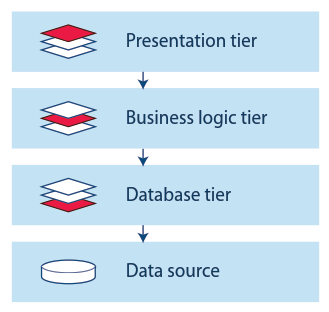
Uma solução de nuvem pública não serve, pois o principal problema é que o ambiente necessita de testes de produção e um fator determinante é que o serviço AWS exige um cartão de crédito para realizar pagamentos. Tal forma de pagamento inviabiliza a utulizanão do serviço, pois a empresa é auditada pelo TCU e o setor que utilizará a solução carece de autonomia, bem como os gestores se negaram a pagar por uma solução que tem o seus custo indefinido e por demanda, atrapalhando o centro de custo e planos orçamentários.

Um grande problema hoje na instituição é que todos os equipamentos estão submetidos a diretivas restritivas de utilização e acesso que impedem os alunos desenvolverem e instalarem aplicativos. Em um ambiente de nuvem privada, a alocação de equipamentos perde sentido, o gerenciamento é centralizado e organizado. A instituição possui hardware de grande capacidade subaproveitado e o custo para o desenvolvimento e aplicação de um IaaS é praticamente nulo.

Para rodar o WordPress, as configurações que a hospedagem deve suportar são:

* Servidor baseado em UNIX/Linux1
* PHP versão 7 ou superior
* MySQL versão 5.6 ou superior OU MariaDB versão 10.0 ou superior
* Memória para o PHP de pelo menos 64 MB (Somente para o software WordPress, sem plugins adicionais)
* 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)

O framework Wordpress é uma aplicação web de 3 camadas que consistem em Apresentação Web, a aplicação e a persistência de base de dados.



Em um cenário antigo, a empresa precisa hospedar em sua estrutura própria equipamentos de hardware que comportem a demanda inferida. Conforme o diagrama abaixo, o investimento necessário para prover o serviço desejado envolve a compra de diversos equipamentos, assim como a compra das licenças dos softwares necessários para a operação da equipe de desenvolvimento. Inclui custos com a manutenção dos equipamentos bem como ao fornecimento de eletricidade e backup em caso de queda de luz. Ao se considerar todos esses custos diretos e ainda os possíveis custos indiretos da implantação e manutenção dos equipamentos se fez necessário pesquisar alternativas menos custosas e mais ágeis para comportar as demandas iniciais do projeto.

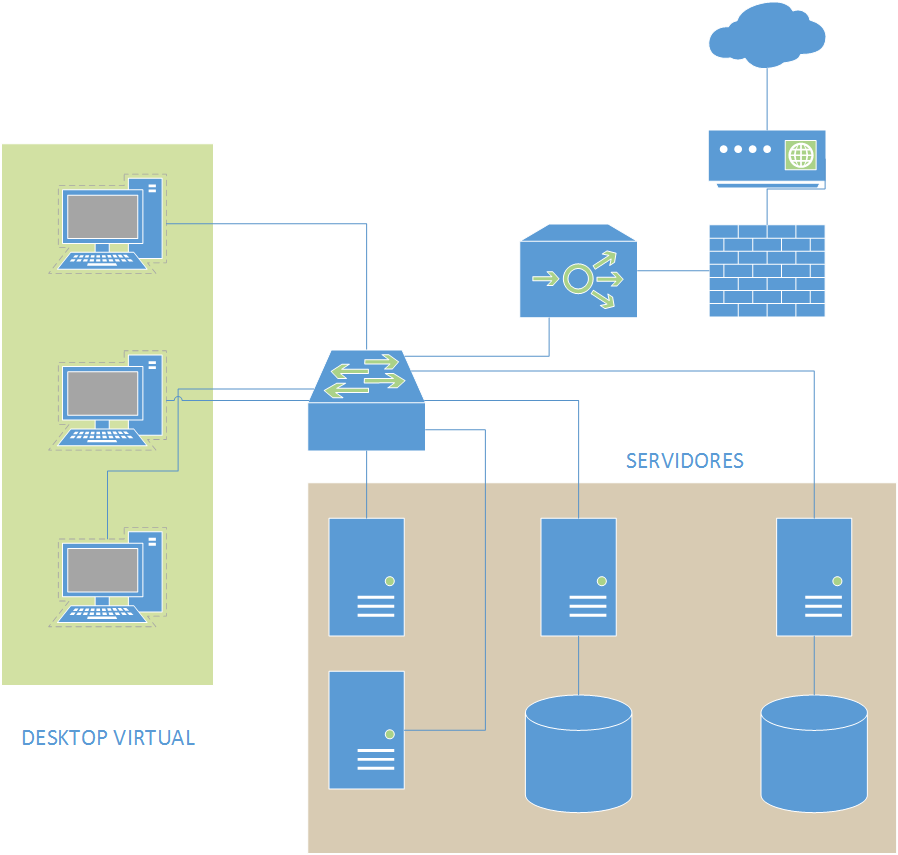
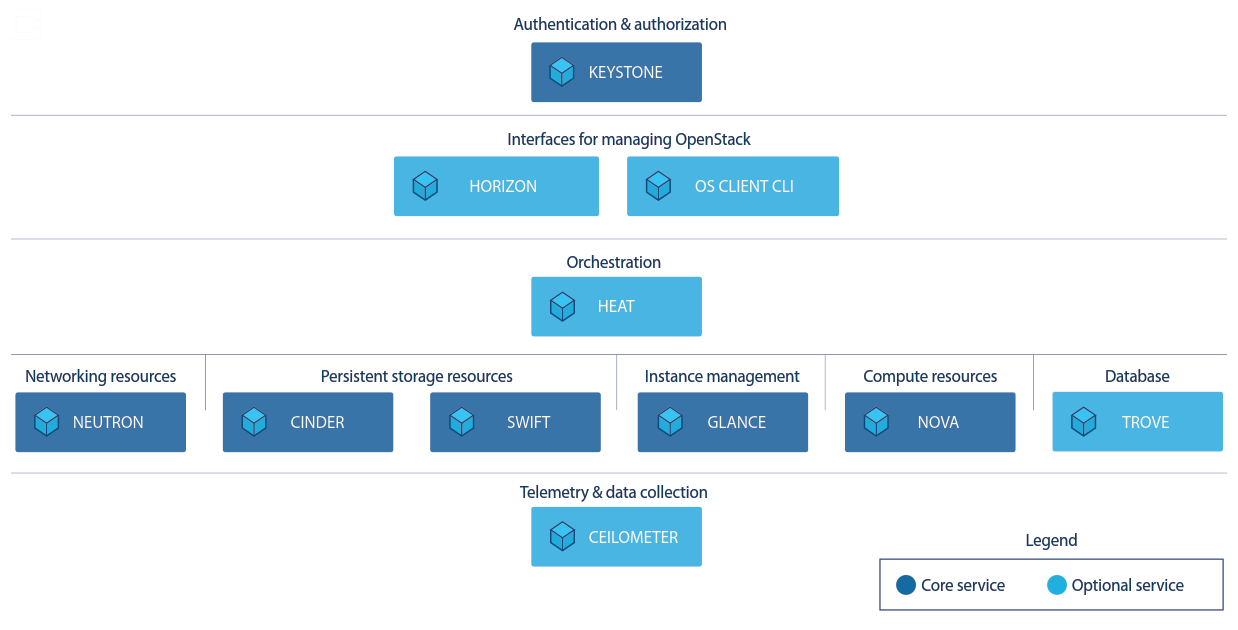


Figura 1 - a topologia física da solução inicial

A aplicação Wordpress necessita ter requisitos específicos que podem ser atendidos com uma solução de nuvem particular baseada na plataforma OpenStack. A plataforma Openstack, dentro do modelo de 3 camadas, pode atender de maneira a auto-escalar os recursos em momentos de pico de acesso, pode prover instalação ágil de instancias, pode prover para cada camada servidores virtuais, pode prover rede e isolamento para ambiente de testes, balanceamento de carga de maneira muito rápida, automatizada em alguns casos e existe ainda a possibilidade de orquestrar a automação de ambientes de desenvolvimento, monitorar o balanceamento de cargas dos servidores e instâncias, pode ainda utilizar a replicação e automação de backups, de maneira que a empresa pode consumir o TI como um serviço rápido e prático.

Existe um ganho notável de eficiência por trás da utilização de uma nuvem privada.

A solução Openstack possui uma organização lógica de seus serviços, conforme orienta o guia de referência da Openstack para arquitetura de aplicações web.



A arquitetura lógica Openstack possui elementos individuais, chamados de projetos, que são ligados de maneira modular, linear e independente. Cada instancia é um compute node individualizado que pode ser conectado ao conjunto de maneira rápida e prática, criando-se um portal de auto-serviço para que os desenvolvedores, por exemplo, criem por conta própria as suas instâncias para realizar os testes sem precisar abrir um chamado. O core e os serviços do Openstack são descritos abaixo.

Computacional (Nova) – Gerencia o ciclo de vida das instancias por completo, incluindo a criação por demanda, agendamentos, e desligamento das instâncias também por demanda.

Serviço de Imagens (Glance) - Armazena e recupera as imagens de discos das máquinas virtuais. É utilizado pelo Nova durante o provisionamento de uma instância

Armazenamento de Bloco (Cinder) Virtualiza o armazenamento de blocos e provê autosserviço para utilizar recursos.

Rede (Neutron) – Habilita a conectividade de rede para todos os serviços Openstack, incluindo as instâncias e a topologia a ser utilizada underlay e overlay através do SDS. Pode prover balanceamento de carga e firewall como serviço.

Serviço de Identidade (Keystone) – provê autenticação e autorização para utilizados os serviços Openstack.

Armazenamento de Objetos (Swift) – Armazena e recupera arbitrariamente dados de objetos sem estrutura, possuindo capacidade de tolerância à falhas e escalonamento.

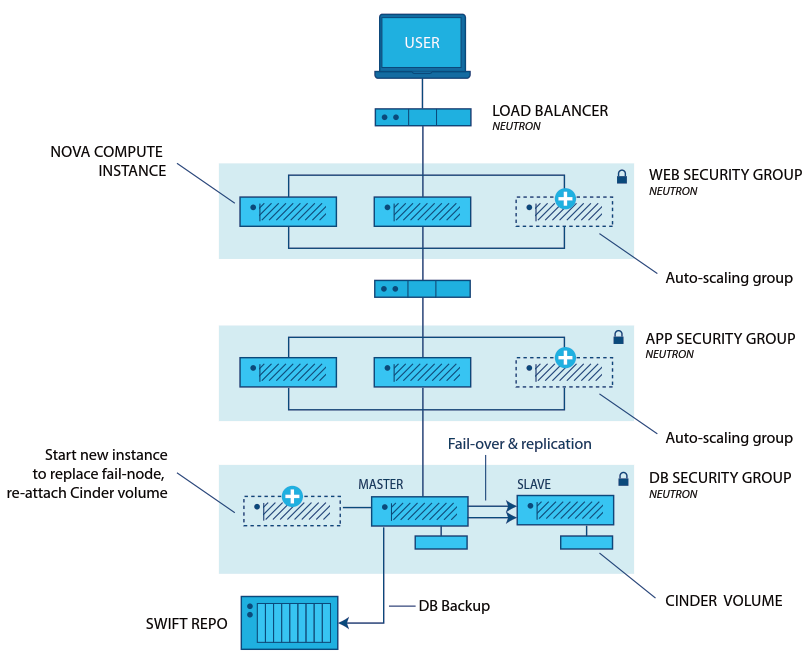
Dashboard (Horizon) – Prove uma interface web para interagir com os serviços do Openstack underlay, como lançar uma instância, prover endereçamento IP, configurar controle de acesso, por exemplo.

Orquestração (Heat) – prove orquestração da automação com base em templates através de sua API.

Telemetria (Ceilometer) – Monitora e mede a nuvem Openstack para permitir cobrança por minutagem ou demanda, benchmarking, escalabilidade e gera estatísticas.

Database (Trove) – possibilita o uso de Database as a service em banco de dados.

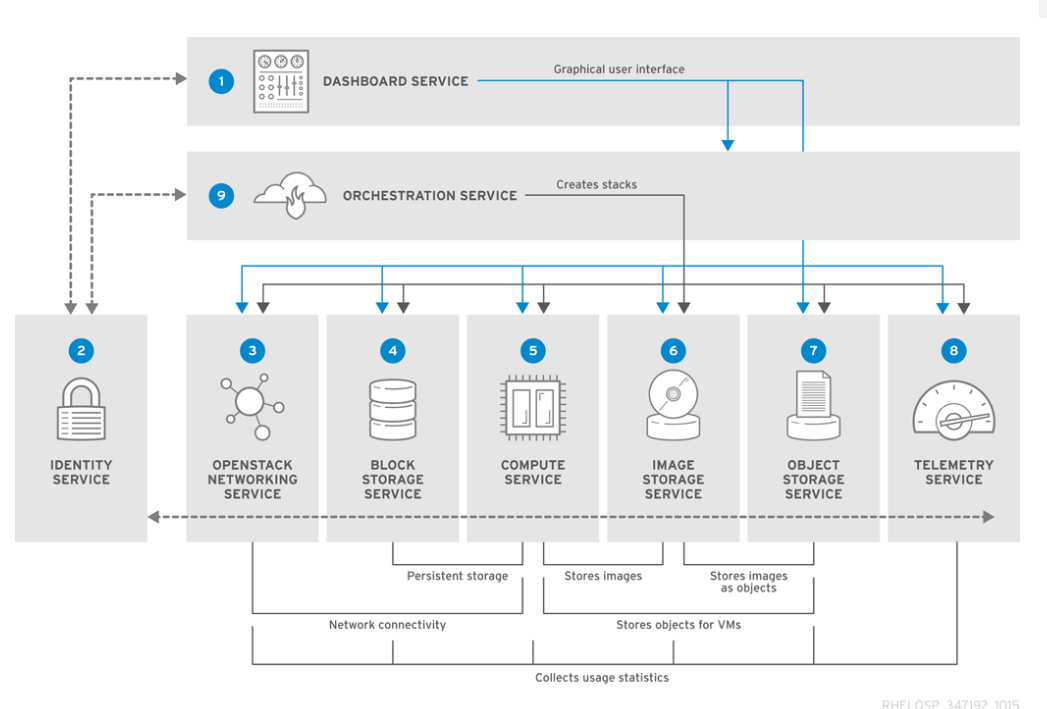
A aplicação de todos esses componentes listados se relaciona na arquitetura da solução de acordo com o diagrama abaixo.



Nessa solução será aplicada a infra com base no Linux Redhat server versão 7.6 Beta1 e será utilizada o agente Ironic Python 13.0 para ambiente de testes e conhecimento da ferramenta inicialmente. Caso os recursos apresentem facilidade de gerenciamento, será em seguida adotado a utilização da solução Red Hat.

Preço da assinatura.

Red Hat Enterprise Linux OpenStack Platform for Controller Nodes Standard: $2,149/socket-pair/year



Softwares que compõem a plataforma:

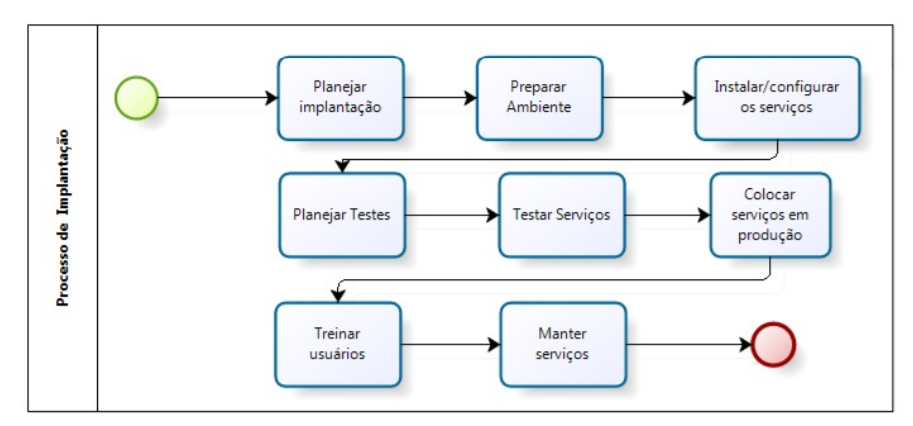
* Red Hat Enterprise Linux (baremetal operating system)
* Red Hat Virtualization Kernelbased Virtual Machine (KVM) Hypervisor
* Red Hat Enterprise Linux High Availability Add-On
* Red Hat OpenStack Platform director
* Red Hat OpenStack Platform
* Red Hat CloudForms
* Red Hat Enterprise Linux (guest operating system)

Plano de Instalação

O plano de instalação seguirá a implementação de short term com o prazo de duração de 3 meses. Durante os 3 meses, serão levantadas as medições abaixo.

* Total vCPU number
* Total vRAM allocation
* I/O mean latency
* Network traffic
* Compute load
* Storage allocation

O fluxo do processo de implantação seguirá o diagrama abaixo



O diagrama de implantação segue o planejamento da implantação, em que todas as premissas detectadas e acordadas no Termo de Abertura do projeto serão estabelecidas. Após o estabelecimento da WBS (Work Breakdown Structure) o ambiente será preparado para receber as imagens. Em seguida os serviços Openstack serão configurados. Após a configuração, os testes para a funcionalidade serão elaborados, serão testados e colocados em produção após as condições dos testes serem satisfeitas. O treinamento para a utilização será realizado e a manutenção dos serviços será realizada, fechando o ciclo do projeto.



Na proposta inicial, na parte dos pré-testes, será colocado um ambiente Openstack completo rodando sobre o Linux Ubuntu. Nesse primeiro momento, toda a implementação será realizada em etapas que incluem:

* descoberta e escolha das versões de software adequadas
* instalação teste da nuvem privada
* documentação da instalação
* backup das imagens testadas
* divulgação da solução
* utilização da solução em turma piloto
* correção e adequação necessária
* levantamento dos custos para migrar o servidor
* implementação de versionamento Github
* obtenção de link externo
* obtenção de infraestrutura para rede sem fio interna

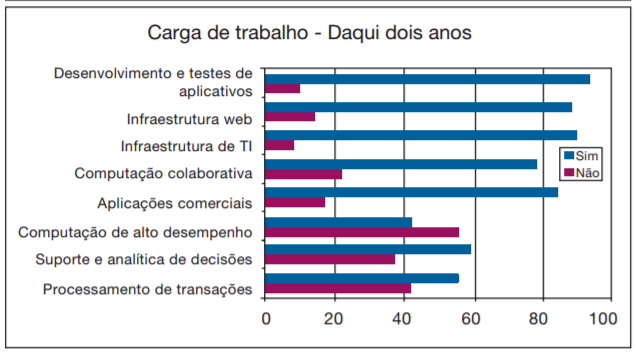
Com essas etapas cumpridas, será realizada uma proposta de implementação de nuvem privada para atender o Fablab da unidade, assim provisionando a solução de virtualização e utilização de Big Data em projetos que atendem a demanda da indústria 4.0

Comparação entre serviços existentes

A título de comparação, a nuvem Openstack foi comparada com a concorrente AWS, nessa comparação alguns fatos foram destacados e listados na tabela abaixo.

|  |  |  |
| --- | --- | --- |
| Características | OPENSTACK | AWS |
| SLA | Suporte Redhat 24/7 | 99,5% |
| Armazenamento | Adaptável | SSD |
| Controle dos seus Dados | Própria Empresa | AWS SLA |
| Custo | Equipe própria de TI e Hardware próprio | Uso por minutagem ou hora ou por bloco de utilização |
| Nuvem pública | Gateway com infra própria | Custo pode subir exponencialmente por demanda exigida |

A comparação do serviço oferecido pela Amazon não se encaixa do modelo de nuvem demandado pela empresa. A Amazon possui forte descrédito, apesar de ser pioneira na solução, por parte dos diretores de TI. No tocante à segurança da informação, ainda é apontada grande preocupação com relação à posse dos dados. Conforme estudo realizado pela IBM, existe uma tendência de se utilizar o ambiente de nuvem privada não para a produção direta, mas para a operação em paralelo para treinamento e testes das equipes que lentamente vão incorporando as funcionalidades mais simples e fáceis de operar, por parte do suporte e por parte dos usuários.



No modelo de trabalho atual da empresa, o Openstack seria mais adequado para a utilização inloco dos colaboradores que possuem a necessidade de rápida disponibilização de ambientes de testes. Nem todos os colaboradores podem utilizar os recursos da nuvem pública por terem dificuldade de obter conexão. É comum usuários perderem os dias trabalhados por terem problemas de segurança, cabos de telefonia furtados impossibilitarem a conexão e dias de trabalho são perdidos dentro do cronograma de desenvolvimento do site.

O Rackspace ofertou uma estrutura que também fugiu do escopo dos custos, já que a empresa possui infra própria e perfeitamente operacional, carecendo apenas de se adptar à demanda dos métodos ágeis de desenvolvimento.

O valor abaixo é em dólares, excluindo taxas e impostos.



Faça upload de uma versão inicial do capítulo 3/Implementação de seu Projeto de Bloco.

Você deve contemplar:

* Referências dos downloads e/ou versões de todos os componentes da solução implementada.
* Todos os passos/capturas de tela de implementação de sua infraestrutura de virtualização.
* Todos os passos/capturas de tela de configuração de sua aplicação.
* Uma captura de tela de sua aplicação em funcionamento.

# 3. Justificativa

O projeto serve para fixar o conhecimento em criação de playbooks, mas acho meio complicado, já que o roteiro de aprendizagem foi feito com base no Ubuntu 14 e Ansible anterior ao 2.11. , conforme o link abaixo indica:

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

O que seria necessário no roteiro de aprendizagem seria atualizar o sistema operacional Ubuntu 16.04 e o Ansible 2.7x.

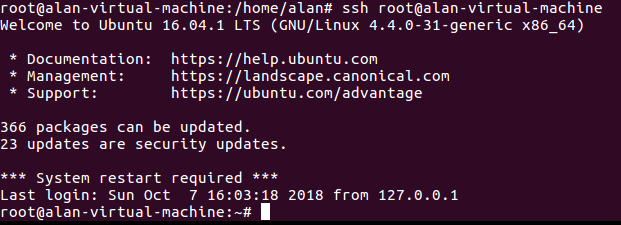
O restante do TP cobriu a publicação do Wordpress e sua automação de instalação no Git.

# 4. Desenvolvimento

# 4.1 Instalação dos pacotes necessários: SSH



Foi criado uma chave para os usuários alan e root e copiados para o servidor para autenticar sem necessidade de senha, garantindo a entrada automaticamente.

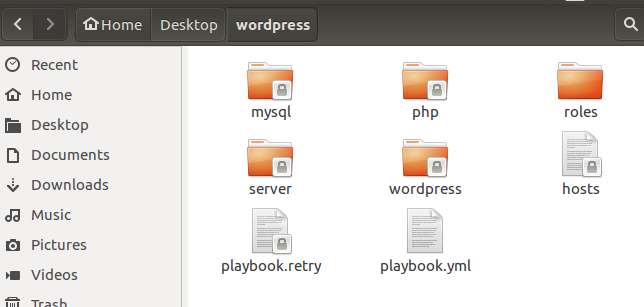


# 4.2 Instalação do Visual Code no Linux.

Foi instalado o Visual Code e o VIM para Ubuntu. No Visual Code foram criados os arquivos necessários para a instalação do Wordpress.

# 4.3 Configuração da estrutura da instalação automatizada

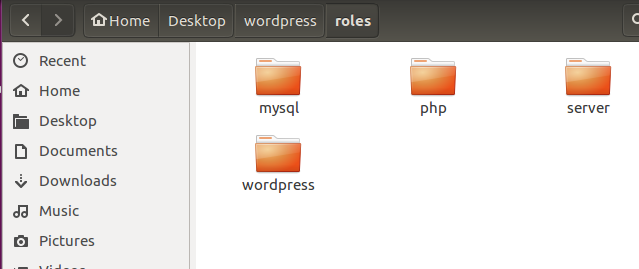
Foi criado uma pasta que contém todas as estruturas que foram criadas e serão armazenadas.



# 4.4 Configuração dos arquivos na estrutura da Automação

Foi criado o arquivo hosts.





Foram baixadas as roles para a criação dos diretórios e subdiretórios necessários para a instalação.

$ cd roles

$ ansible-galaxy init server

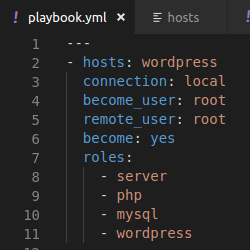
$ ansible-galaxy init php

$ ansible-galaxy init mysql

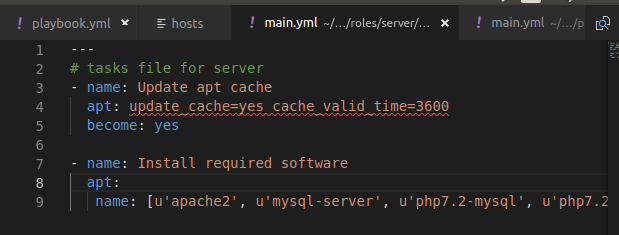
$ ansible-galaxy init wordpress

# 4.5 Criação dos playbooks.

Foi criado o playbook.yml que ficará na pasta raiz do Wordpress

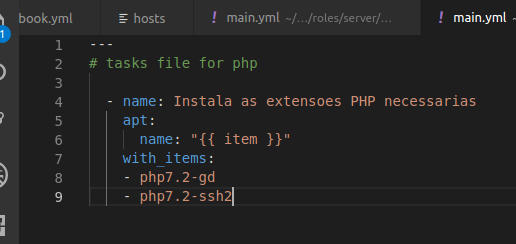


Foi criada a role para instalação dos pacotes no servidor. Essa criação ficou muito controversa, pois os comandos fornecidos para a instalação em loop foram deprecados e a montagem dessa linha de comando ficou quase impossível de re resolver sem orientação do docente.



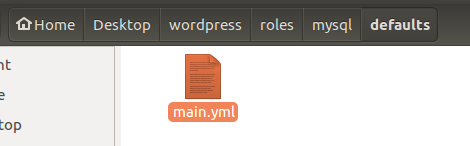
# 4.6 Configuração das tasks do PHP

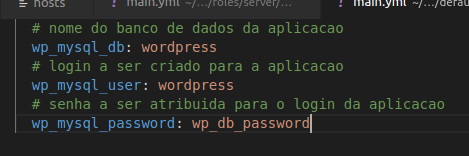
Foram criadas as tasks para o PHP

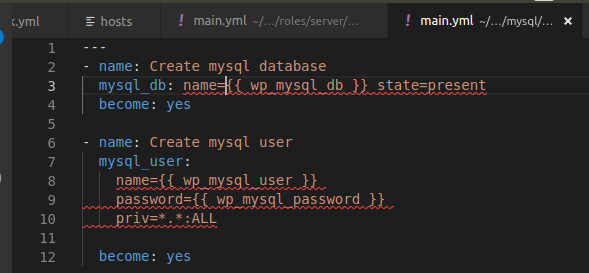


# 4.7 Configuração do Mysql

Foram ajustadas as informações do arquivo main para a criação do banco de dados da aplicação.



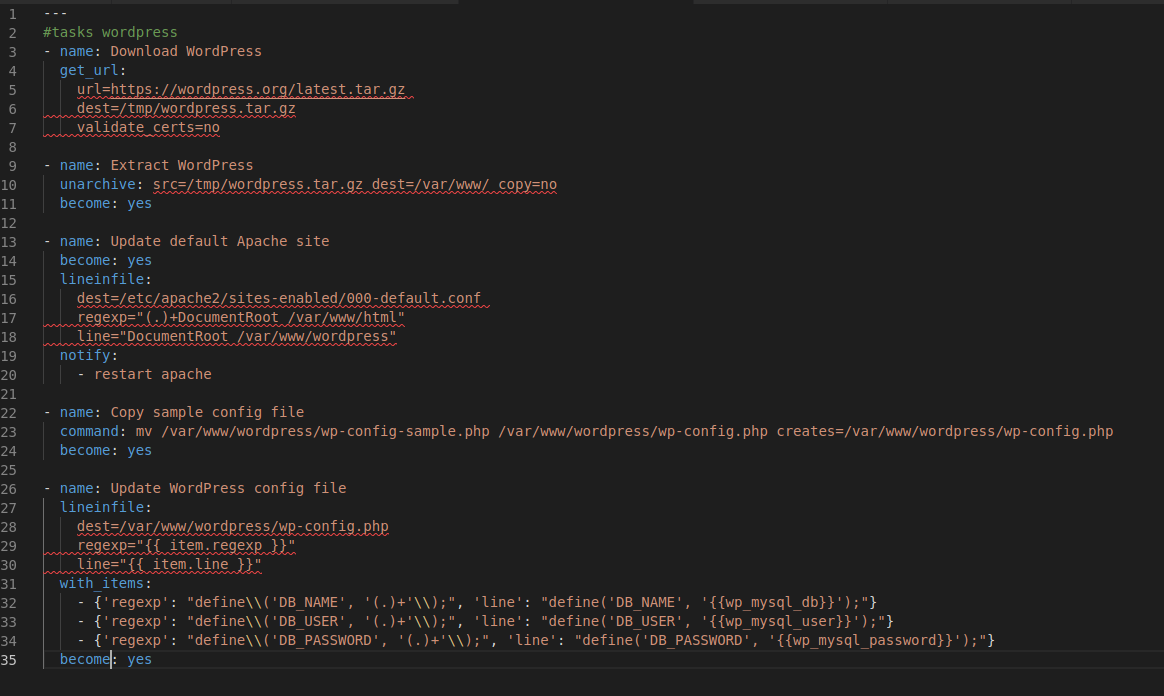




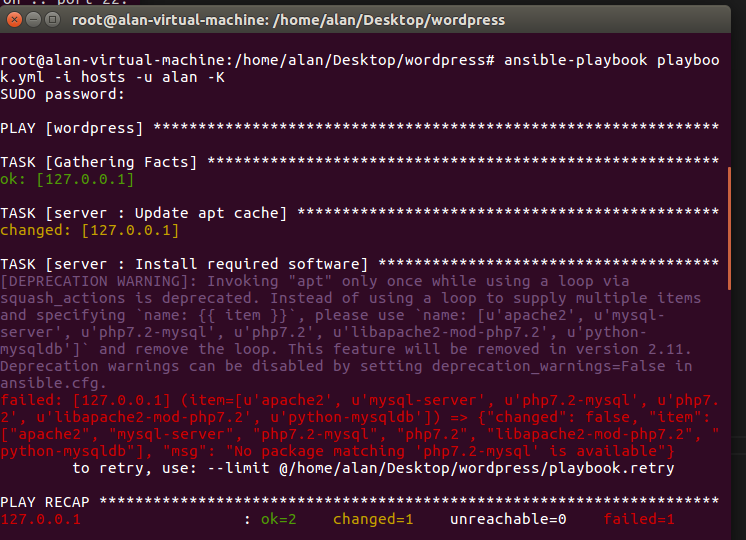
Como pode se observar na foto, existem muitos erros de sintaxe que não puderam ser consertados por falta de entendimento e atualização do roteiro de aprendizagem.

# 4.8 Configuração da task final de Configuração dos pacotes instalados.

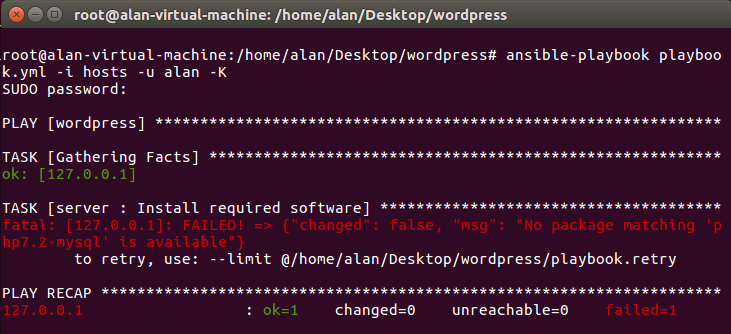
A task final para a instalação consiste na imagem abaixo:



Infelizmente, a composição dos comandos não obteve um resultado satisfatório, conforme a imagem abaixo:

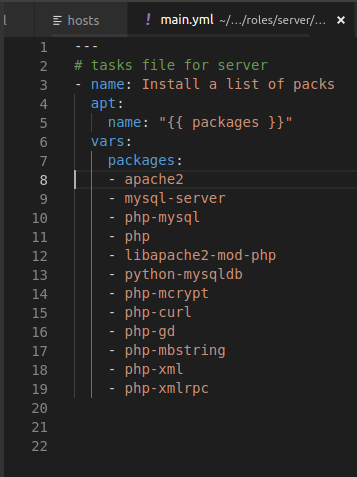


Novamente a adequação do código não foi satisfatório.

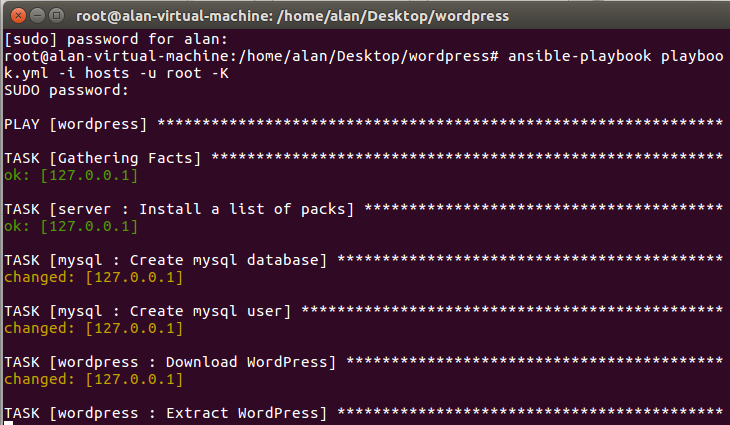


Os pacotes de instalação possuem o nome correto, porém a sintaxe do loop não é mais suportada, quando é feito squash.

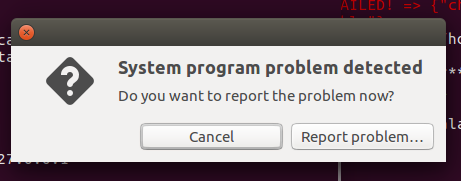
Após intenso estudo foi possível buscar os nomes corretos de cada pacote necessário no repositório e foram testadas e consertadas por execução cada um deles, identados e preenchidos da maneira correta para a versão do Ansible 2.7.



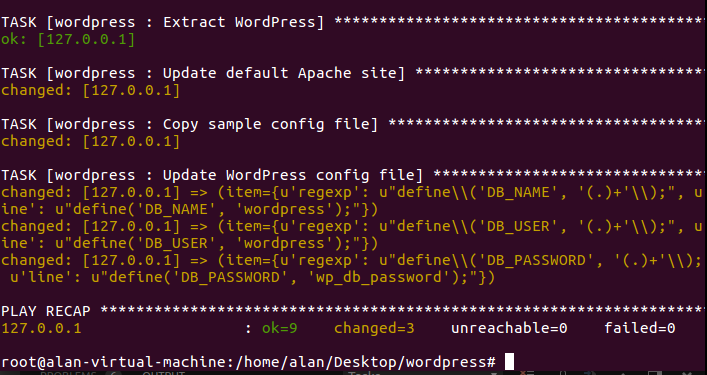
O código apresentou sua execução aceita pelo Ansible.



Mesmo após todas as correções nas linhas de código, o sistema operacional apresentou travamentos.



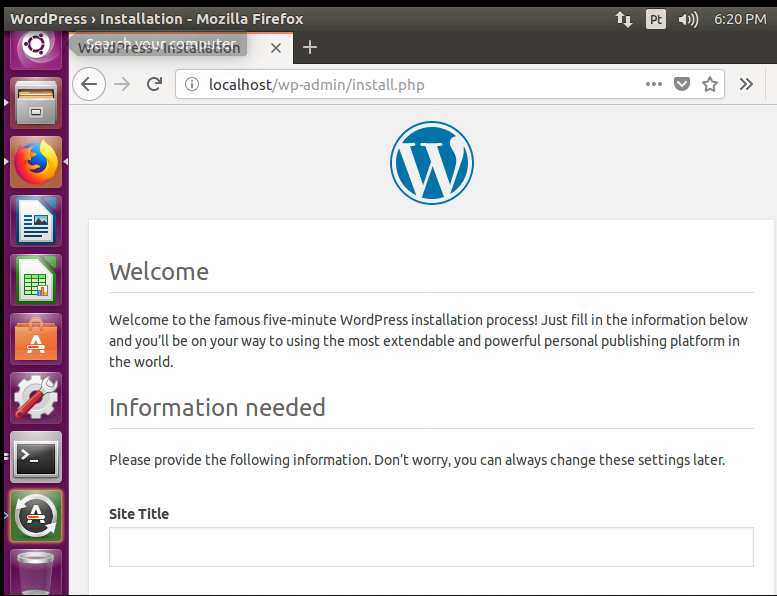
Update no Python e instalado e executado como root, teve os problemas resolvidos.



O servidor apache foi reiniciado.

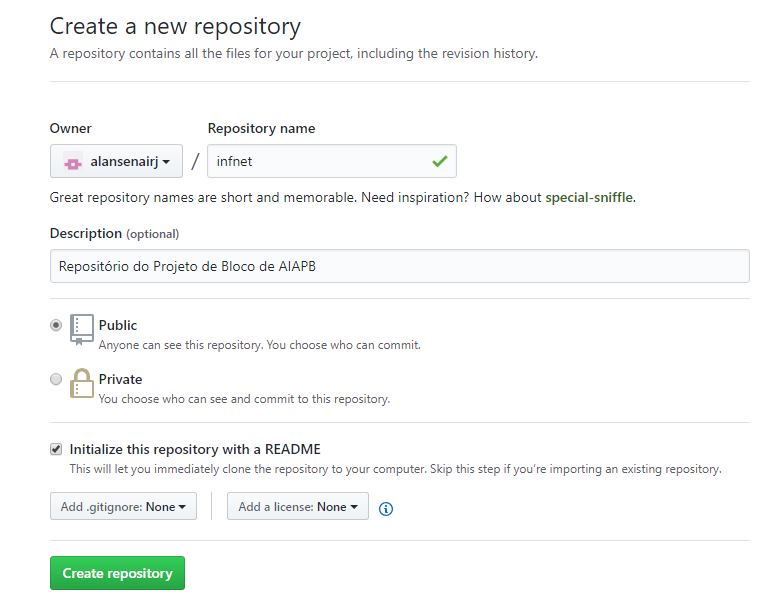


A aplicação está operacional.



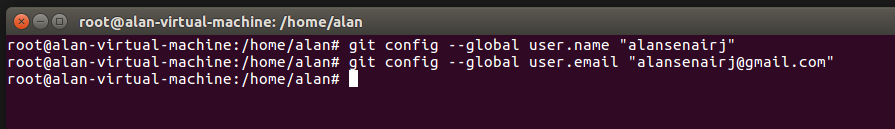
# 4.9 Criação de Repositório no Git

Foi criado um repositório no Git para abrigar os arquivos necessários para a automação da instalação do Wordpress.



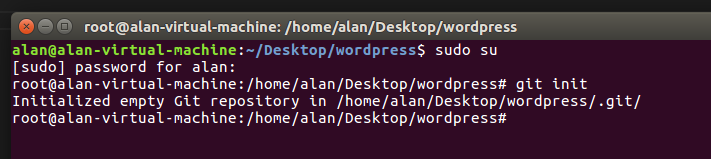
# 4.10 Configuração do Git

Foram configurados no Git o nome do usuário e o e-mail.



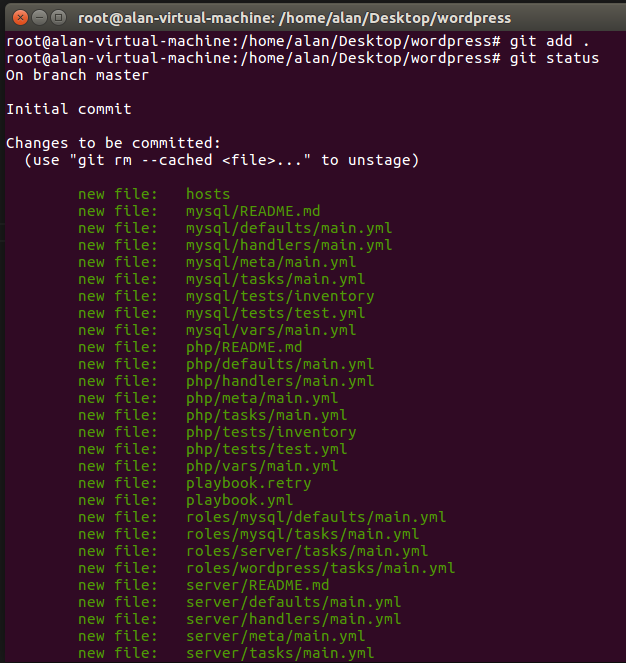
# 4.11 Inicialização do Git

Foi realizada a inicialização do Git, que consiste na cópia de um diretório oculto em que esse repositório copia os arquivos do projeto e os envia para o repositório.



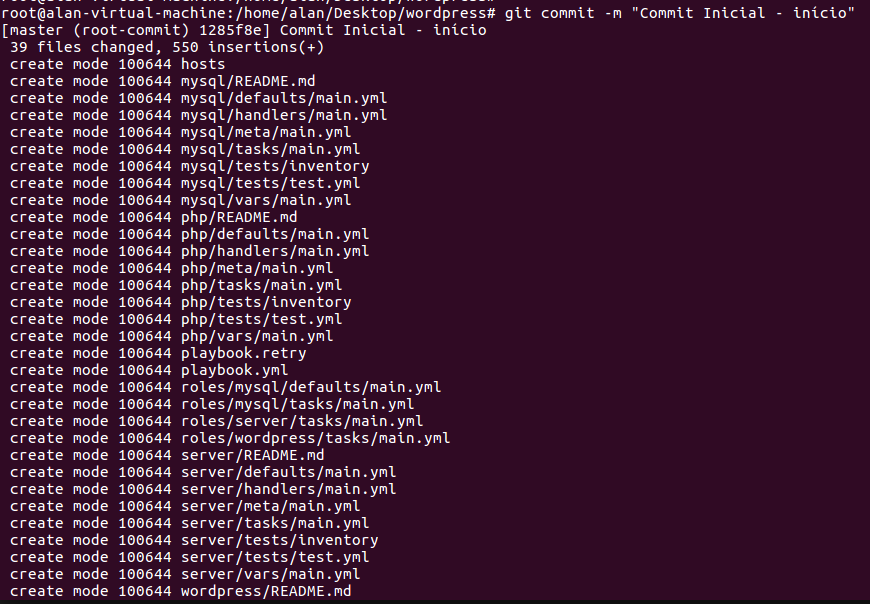
# 4.12 Git Add

Foram adicionados os arquivos da automação da instalação do Wordpress.



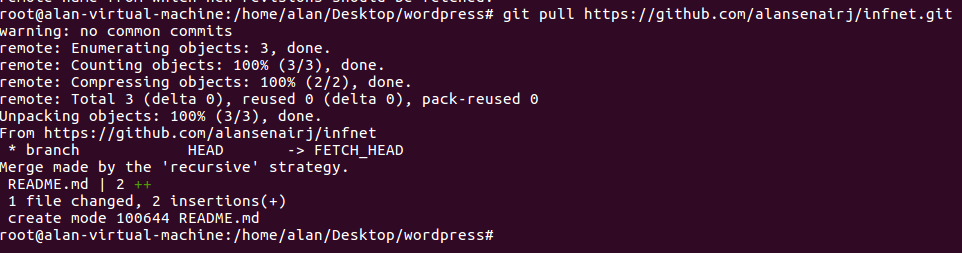
# 4.13 Commit inicial

Foi realizado o commit inicial para marcar onde o projeto original começou.



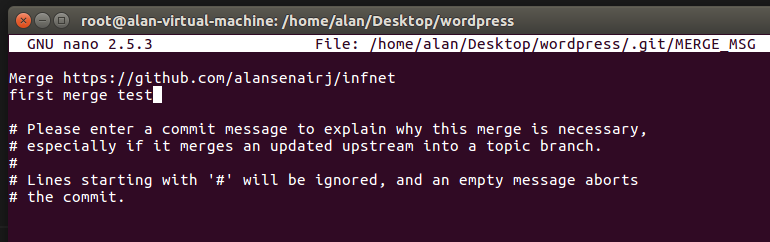
# 4.14 Pull realizado

Foi realizado um pull para testar o versionamento do Git.



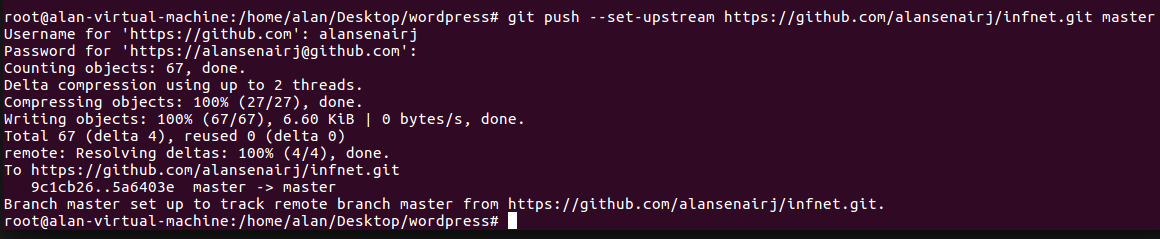
# 4.15 Detalhes do Merge

Ao realizar o pull, o merge solicita mensagens de controle de versão.



# 4.16 Push Realizado

Foi realizado um push do projeto para o repositório.

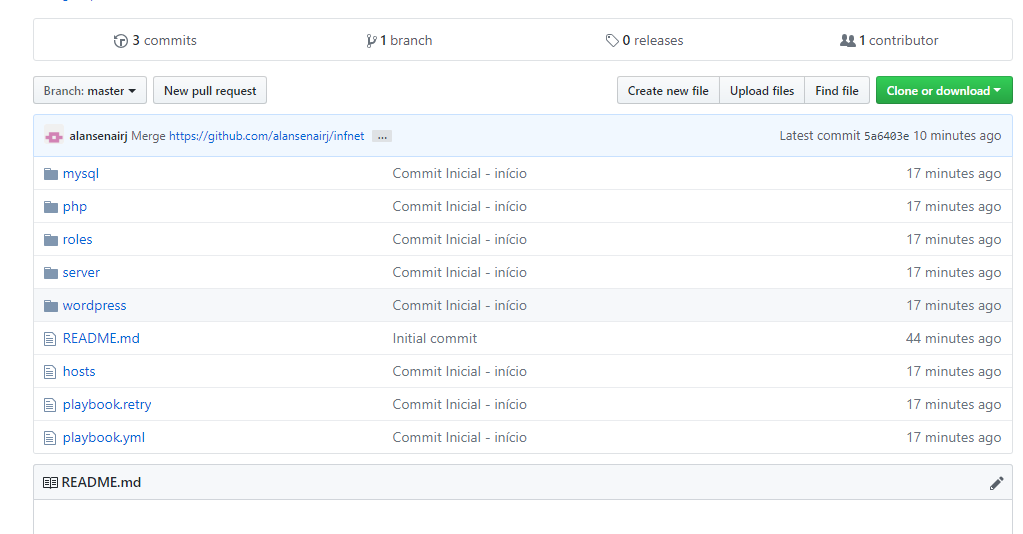


As URLs disponibilizadas são:

<https://github.com/alansenairj/infnet.git>

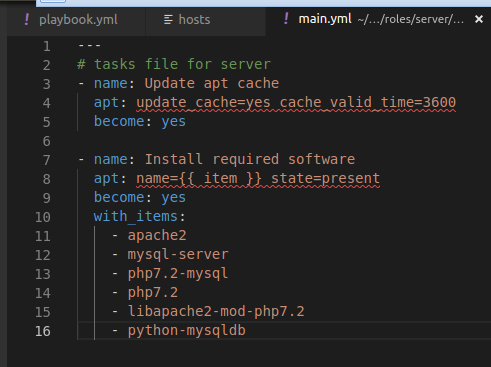
<https://github.com/alansenairj/infnet>

A imagem abaixo exibe o projeto inserido no repositório.



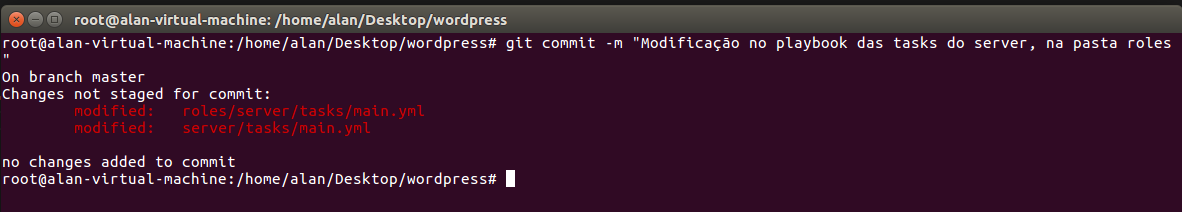
# 4.17 Alteração no código

Foram realizadas alterações no playbook.yml



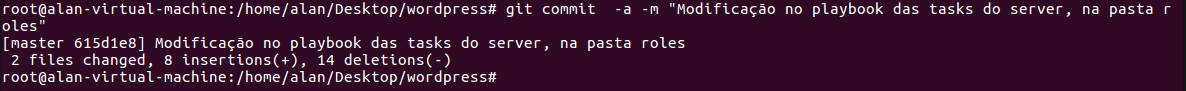
# 4.18 Git Commit no playbook

Foi realizado um commit das alterações.



# 4.19 Stage incorporado

As alterações foram inseridas no repositório.



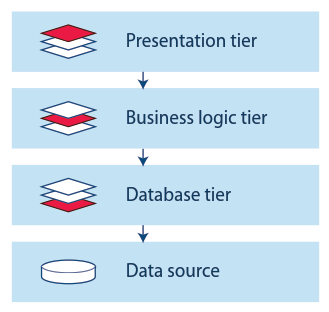
A empresa necessita criar um blog utilizando o framework Wordpress. Essa empresa necessita das seguintes atribuições de seu TI

* Ambiente de desktops virtuais para criação do site
* Ambiente de produção separado de ambiente de testes
* Armazenamento de seu banco de dados
* Backup e redundância
* Performance elástica das instâncias para dias de forte demanda de acesso
* Balanceamento de carga para lidar com dias de forte demanda de acesso
* Firewall e roteamento

Para rodar o WordPress, as configurações que a hospedagem deve suportar são:

* Servidor baseado em UNIX/Linux1
* PHP versão 7 ou superior
* MySQL versão 5.6 ou superior OU MariaDB versão 10.0 ou superior
* Memória para o PHP de pelo menos 64 MB (Somente para o software WordPress, sem plugins adicionais)
* 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)

O framework Wordpress é uma aplicação web de 3 camadas que consistem em Apresentação Web, a aplicação e a persistência de base de dados.



Em um cenário antigo, a empresa precisa hospedar em sua estrutura própria equipamentos de hardware que comportem a demanda inferida. Conforme o diagrama abaixo, o investimento necessário para prover o serviço desejado envolve a compra de diversos equipamentos, assim como a compra das licenças dos softwares necessários para a operação da equipe de desenvolvimento. Inclui custos com a manutenção dos equipamentos bem como ao fornecimento de eletricidade e backup em caso de queda de luz. Ao se considerar todos esses custos diretos e ainda os possíveis custos indiretos da implantação e manutenção dos equipamentos se fez necessário pesquisar alternativas menos custosas e mais ágeis para comportar as demandas iniciais do projeto.

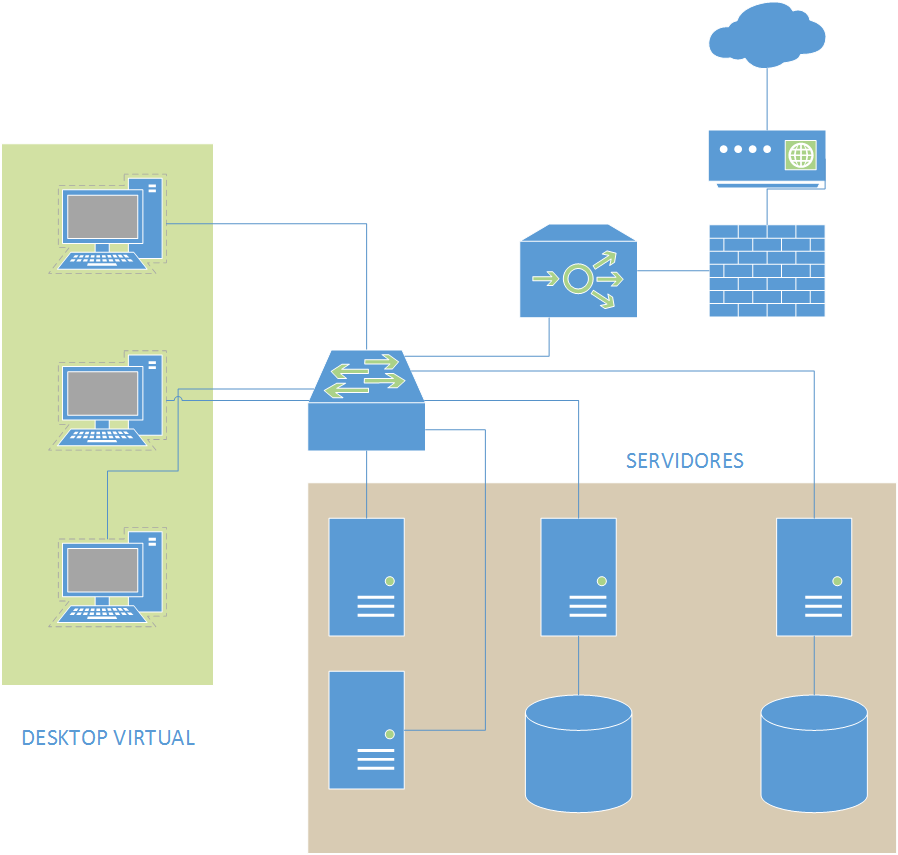
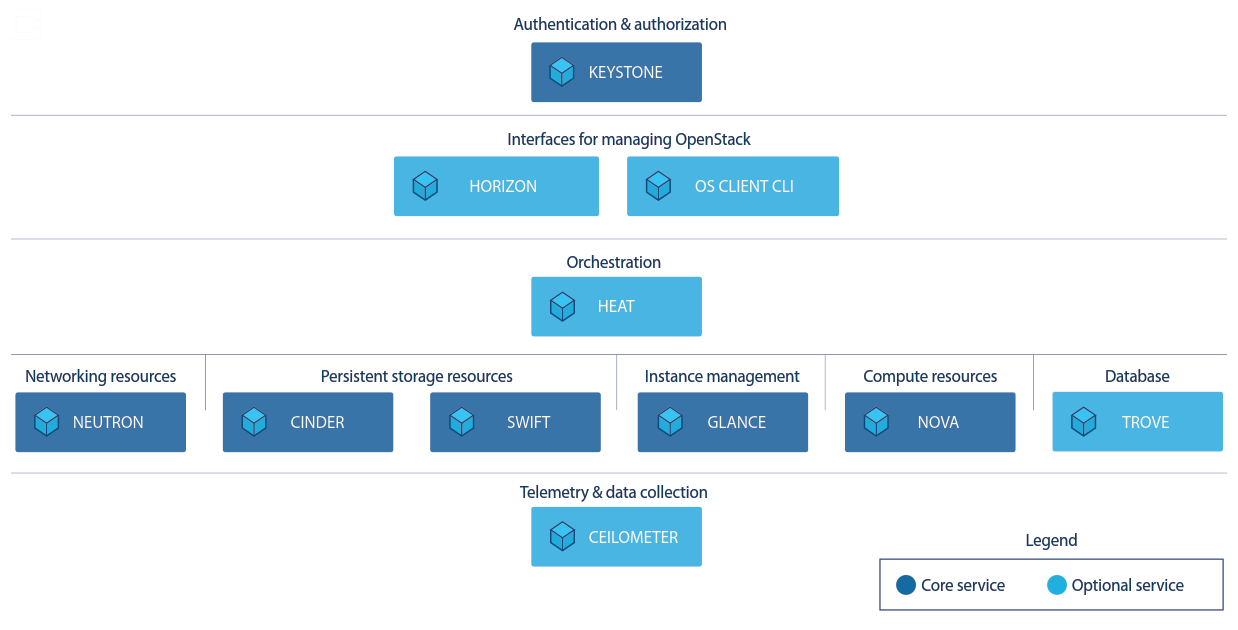


Figura 1 - a topologia física da solução inicial

A aplicação Wordpress necessita ter requisitos específicos que podem ser atendidos com uma solução de nuvem particular baseada na plataforma OpenStack. A plataforma Openstack, dentro do modelo de 3 camadas, pode atender de maneira a auto-escalar os recursos em momentos de pico de acesso, pode prover instalação ágil de instancias, pode prover para cada camada servidores virtuais, pode prover rede e isolamento para ambiente de testes, balanceamento de carga de maneira muito rápida, automatizada em alguns casos e existe ainda a possibilidade de orquestrar a automação de ambientes de desenvolvimento, monitorar o balanceamento de cargas dos servidores e instâncias, pode ainda utilizar a replicação e automação de backups, de maneira que a empresa pode consumir o TI como um serviço rápido e prático.

Existe um ganho notável de eficiência por trás da utilização de uma nuvem privada.

A solução Openstack possui uma organização lógica de seus serviços, conforme orienta o guia de referencia da Openstack para arquitetura de aplicações web.



A arquitetura lógica Openstack possui elementos individuais, chamados de projetos, que são ligados de maneira modular, linear e independente. Cada instancia é um compute node individualizado que pode ser conectado ao conjunto de maneira rápida e prática, criando-se um portal de auto-serviço para que os desenvolvedores, por exemplo, criem por conta própria as suas instâncias para realizar os testes sem precisar abrir um chamado. O core e os serviços do Openstack são descritos abaixo.

Computacional (Nova) – Gerencia o ciclo de vida das instancias por completo, incluindo a criação por demanda, agendamentos, e desligamento das instâncias também por demanda.

Serviço de Imagens (Glance) - Armazena e recupera as imagens de discos das máquinas virtuais. É utilizado pelo Nova durante o provisionamento de uma instância

Armazenamento de Bloco (Cinder) Virtualiza o armazenamento de blocos e provê autosserviço para utilizar recursos.

Rede (Neutron) – Habilita a conectividade de rede para todos os serviços Openstack, incluindo as instâncias e a topologia a ser utilizada underlay e overlay através do SDS. Pode prover balanceamento de carga e firewall como serviço.

Serviço de Identidade (Keystone) – provê autenticação e autorização para utilizados os serviços Openstack.

Armazenamento de Objetos (Swift) – Armazena e recupera arbitrariamente dados de objetos sem estrutura, possuindo capacidade de tolerância à falhas e escalonamento.

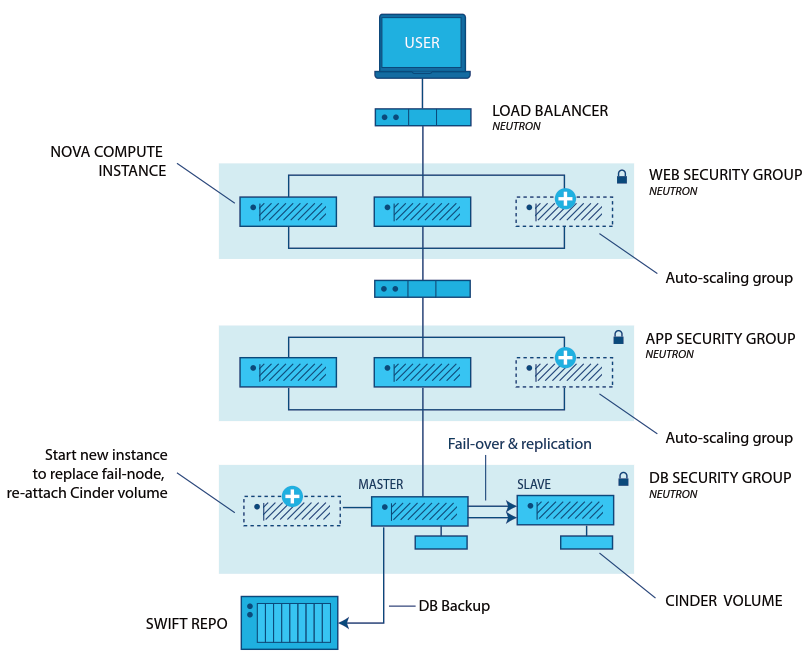
Dashboard (Horizon) – Prove uma interface web para interagir com os serviços do Openstack underlay, como lançar uma instância, prover endereçamento IP, configurar controle de acesso, por exemplo.

Orquestração (Heat) – prove orquestração da automação com base em templates através de sua API.

Telemetria (Ceilometer) – Monitora e mede a nuvem Openstack para permitir cobrança por minutagem ou demanda, benchmarking, escalabilidade e gera estatísticas.

Database (Trove) – possibilita o uso de Database as a service em banco de dados.

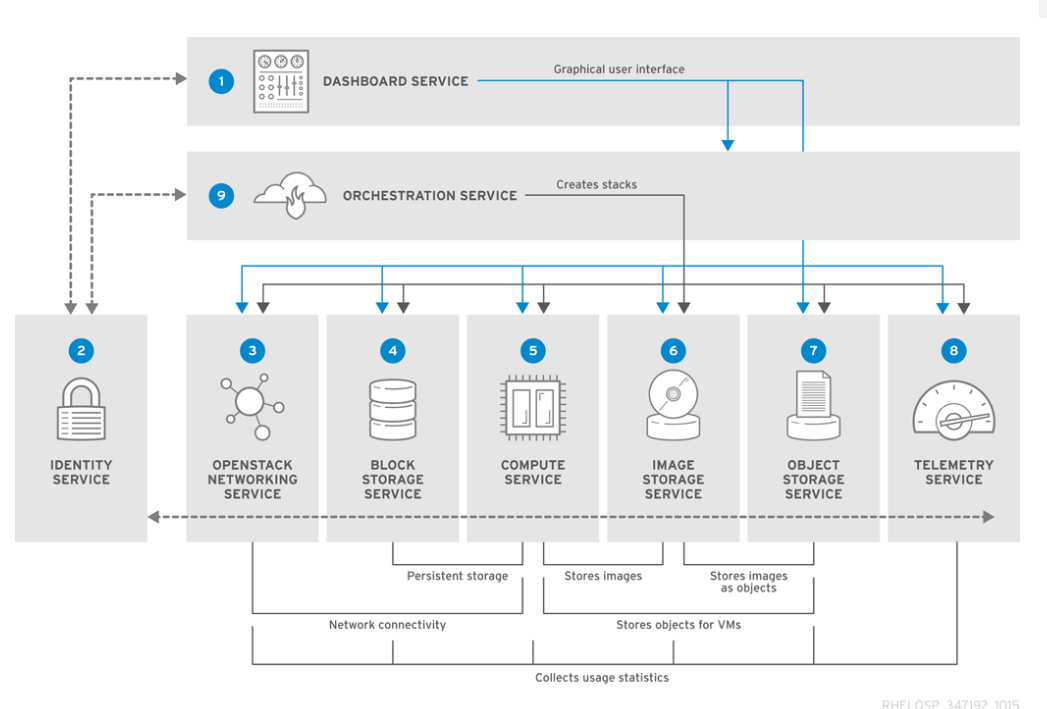
A aplicação de todos esses componentes listados se relaciona na arquitetura da solução de acordo com o diagrama abaixo.



Nessa solução será aplicada a infra com base no Linux Redhat server versão 7.6 Beta1 e será utilizada o agente Ironic Python 13.0 para ambiente de testes e conhecimento da ferramenta inicialmente. Caso os recursos apresentem facilidade de gerenciamento, será em seguida adotado a utilização da solução Red Hat.

Preço da assinatura.

Red Hat Enterprise Linux OpenStack Platform for Controller Nodes Standard: $2,149/socket-pair/year



Softwares que compõem a plataforma:

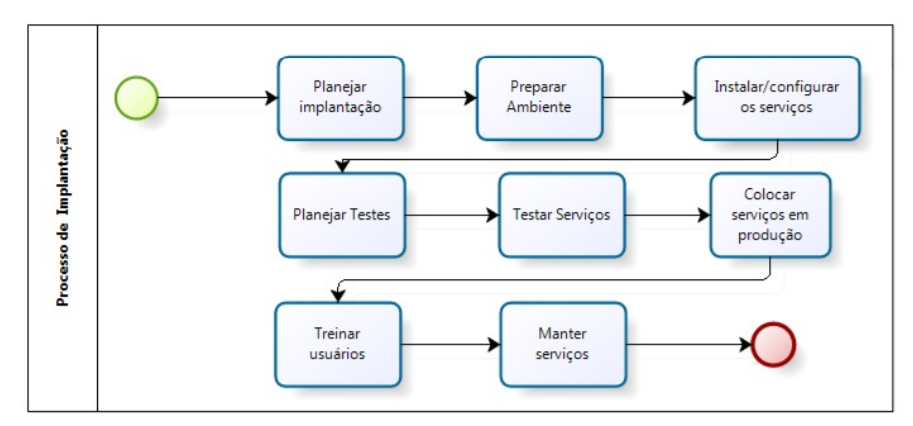
* Red Hat Enterprise Linux (baremetal operating system)
* Red Hat Virtualization Kernelbased Virtual Machine (KVM) Hypervisor
* Red Hat Enterprise Linux High Availability Add-On
* Red Hat OpenStack Platform director
* Red Hat OpenStack Platform
* Red Hat CloudForms
* Red Hat Enterprise Linux (guest operating system)

Plano de Instalação

O plano de instalação seguirá a implementação de short term com o prazo de duração de 3 meses. Durante os 3 meses, serão levantadas as medições abaixo.

* Total vCPU number
* Total vRAM allocation
* I/O mean latency
* Network traffic
* Compute load
* Storage allocation

O fluxo do processo de implantação seguirá o diagrama abaixo



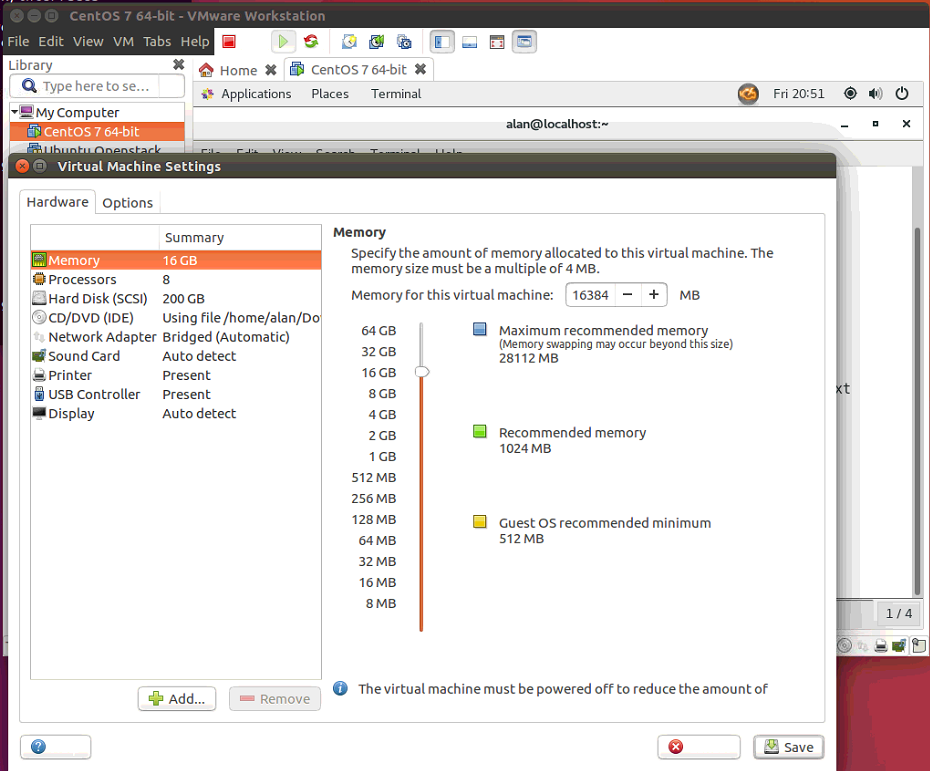
O diagrama de implantação segue o planejamento da implantação, em que todas as premissas detectadas e acordadas no Termo de Abertura do projeto serão estabelecidas. Após o estabelecimento da WBS (Work Breakdown Structure) o ambiente será preparado para receber as imagens. Em seguida os serviços Openstack serão configurados. Após a configuração, os testes para a funcionalidade serão elaborados, serão testados e colocados em produção após as condições dos testes serem satisfeitas. O treinamento para a utilização será realizado e a manutenção dos serviços será realizada, fechando o ciclo do projeto.

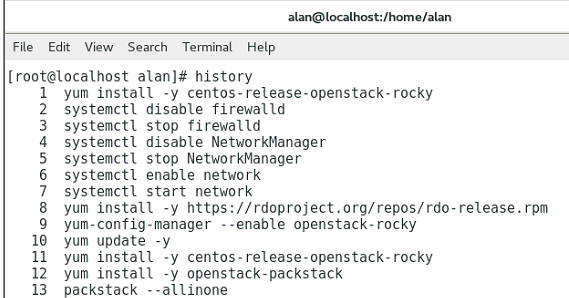


## 2.9 Instalação sob CentOS e Packstack

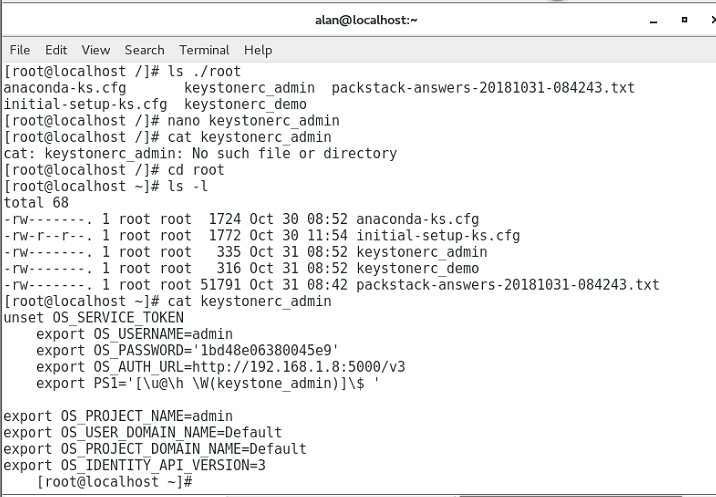
A única opção que funcionou foi a combinação de VMWare WS 15 + CentOS 7 + Packstack.

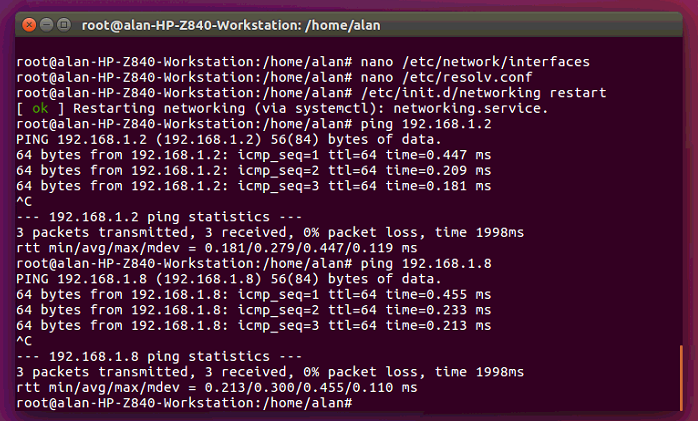
Apesar do Packstack não ser uma instalação completa e indicada para a produção, foi a que conseguiu ser instalada com relativa facilidade e operou no console do Openstack sem apresentar problemas.

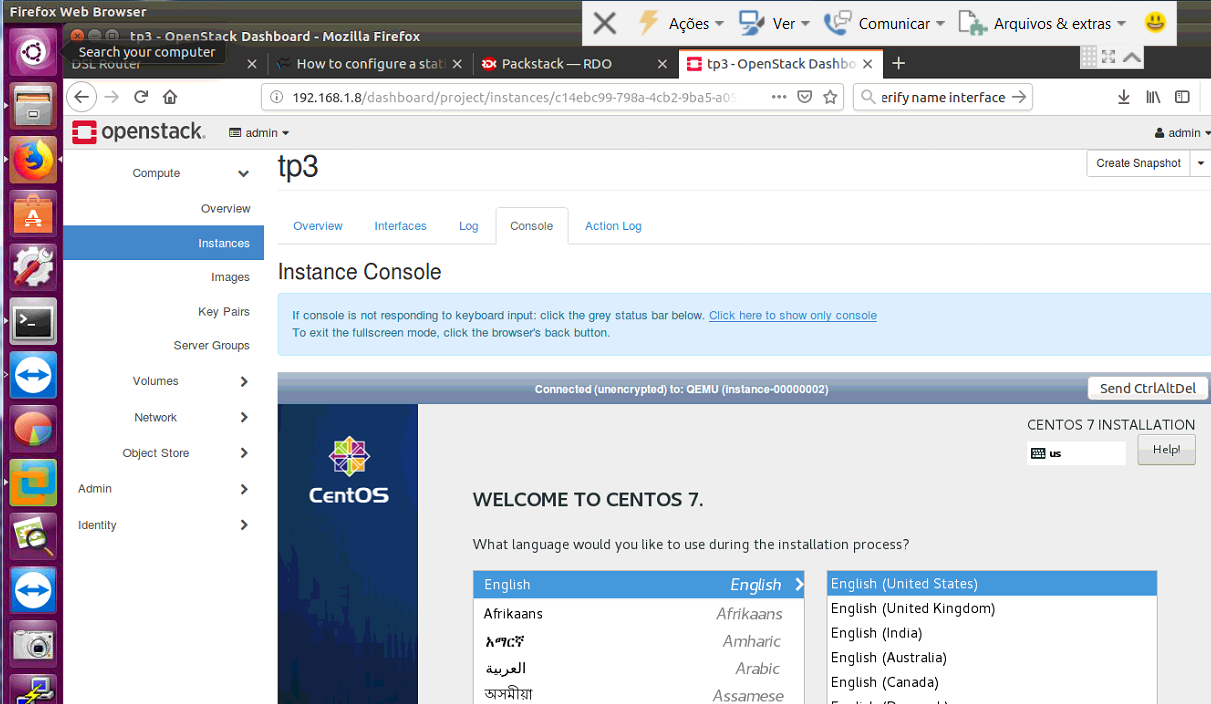




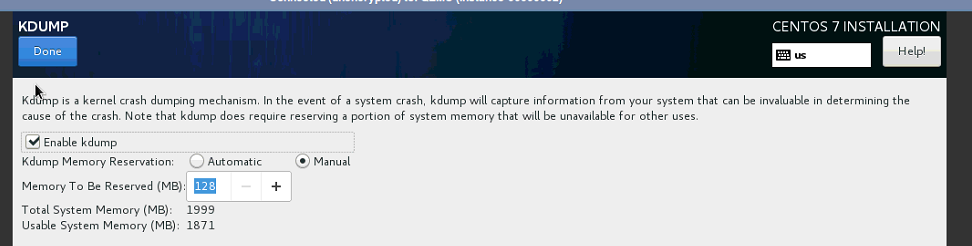


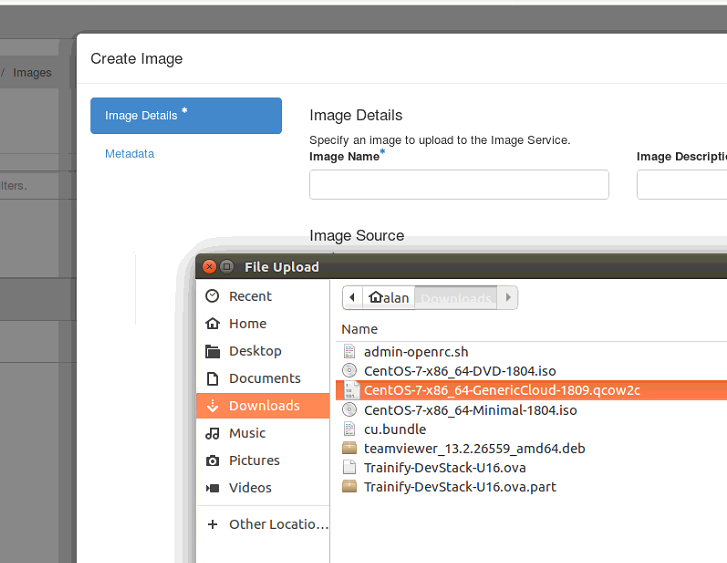




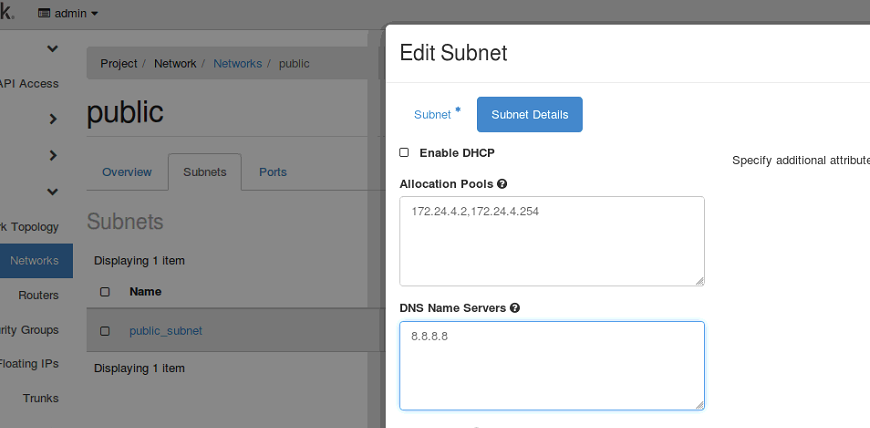


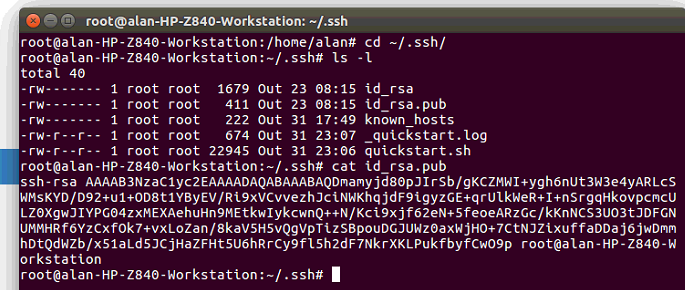
Excessiva lentidão ocasiona erros inesperados no processo de instalação

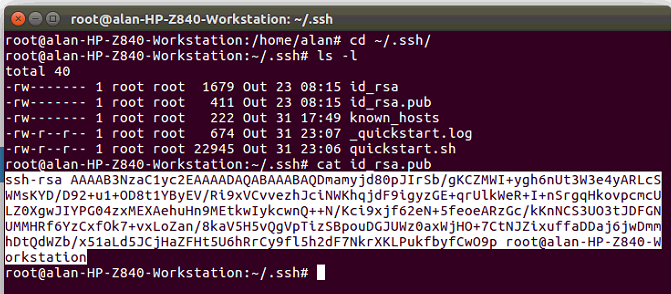


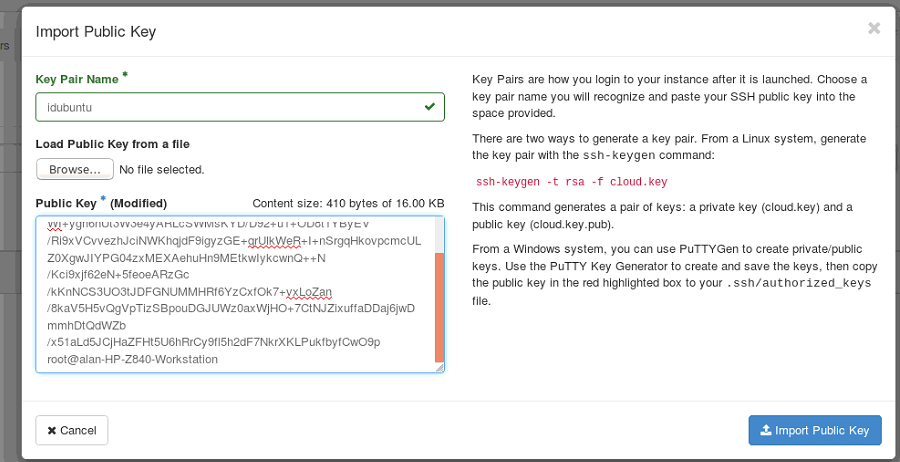


https://cloud.centos.org/centos/7/images/CentOS-7-x86\_64-GenericCloud-1809.qcow2c

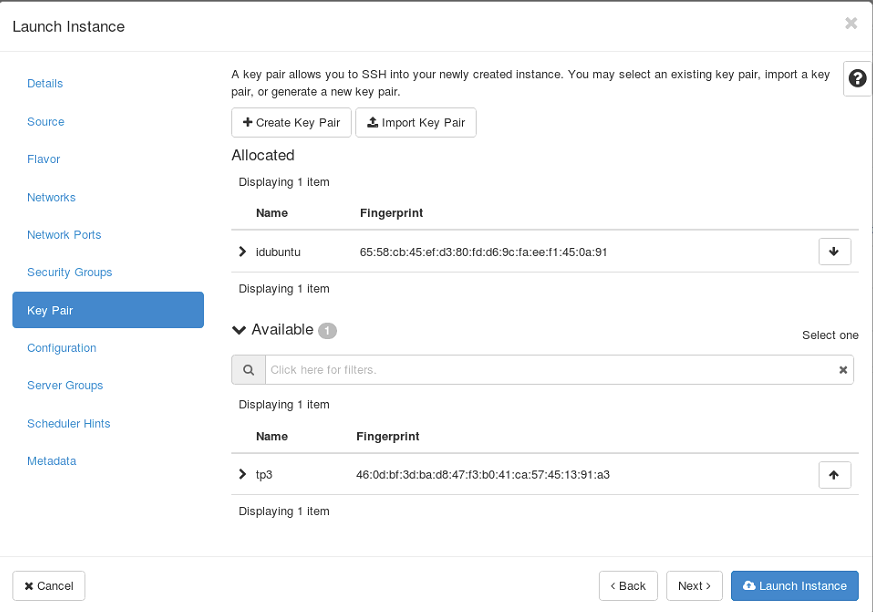




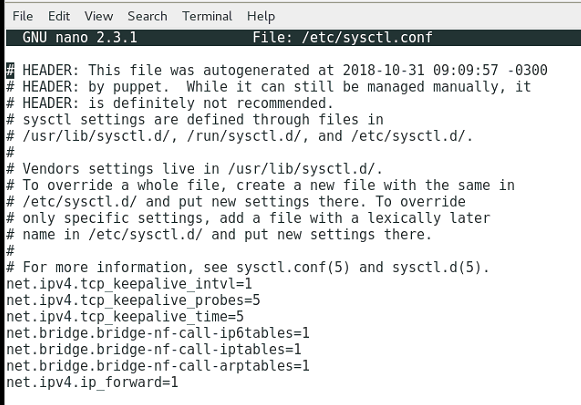


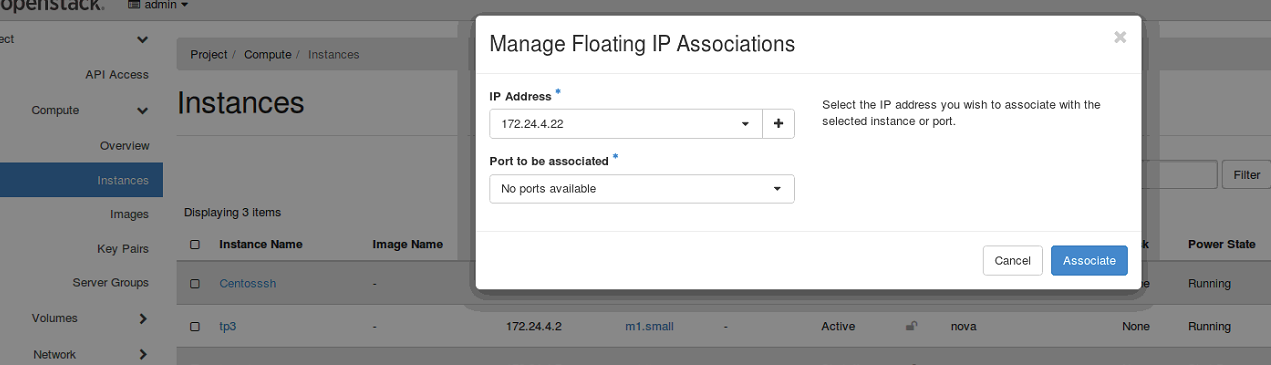




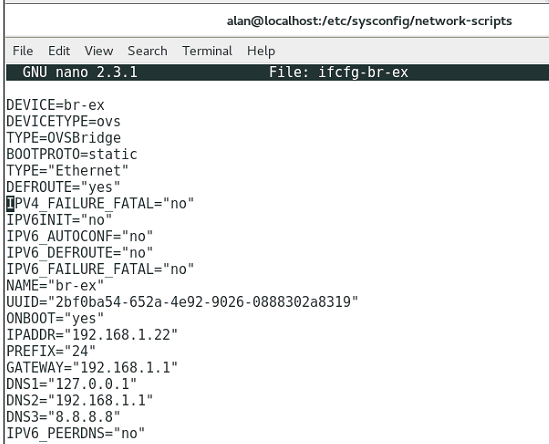






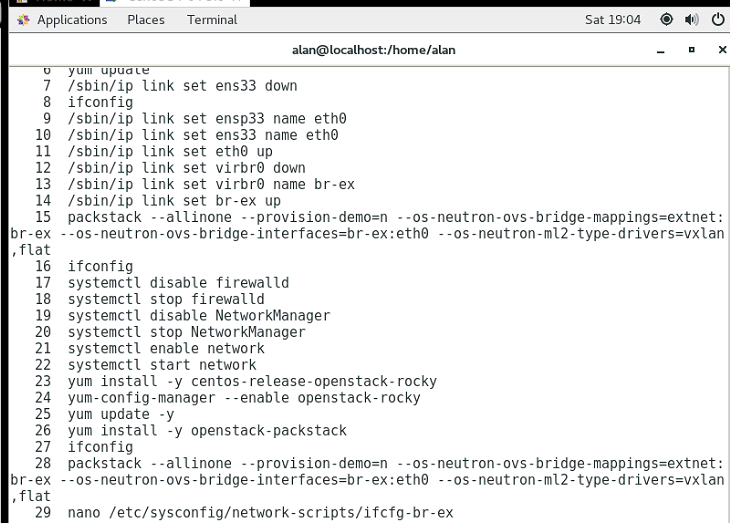


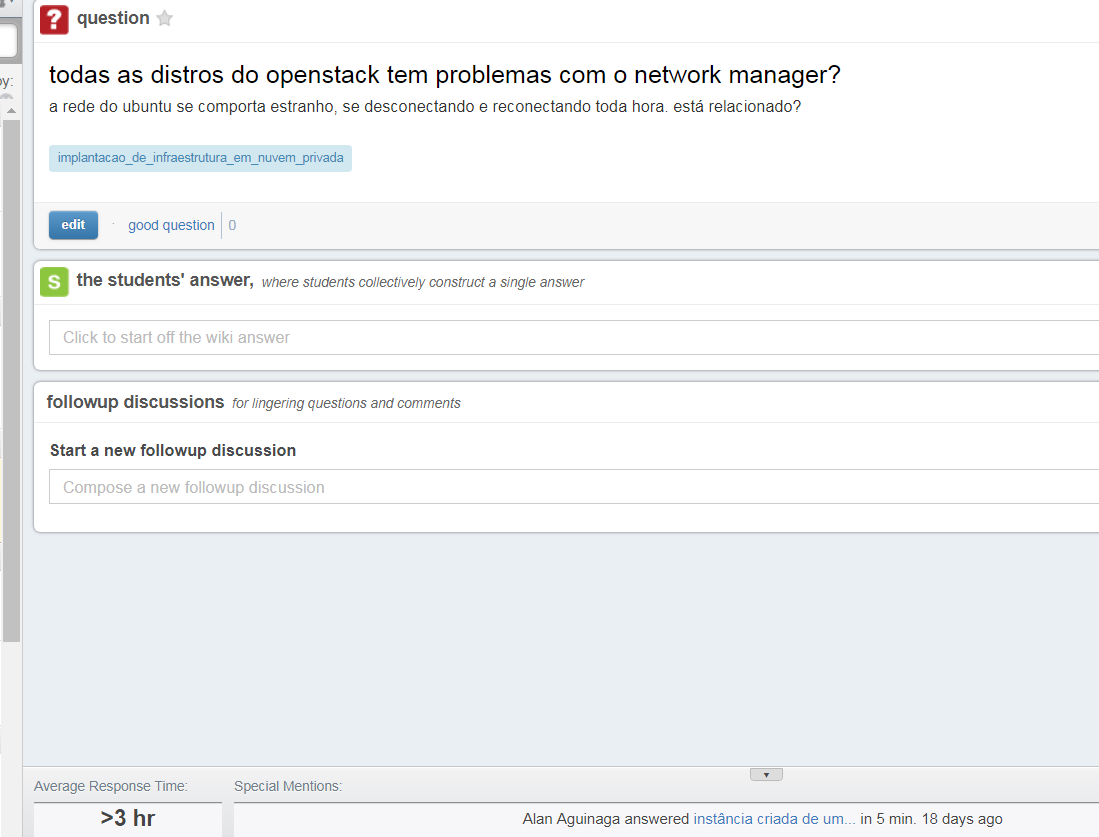




## 2.10 Captura de tela de um acesso à sua VM via SSH ou RDP.

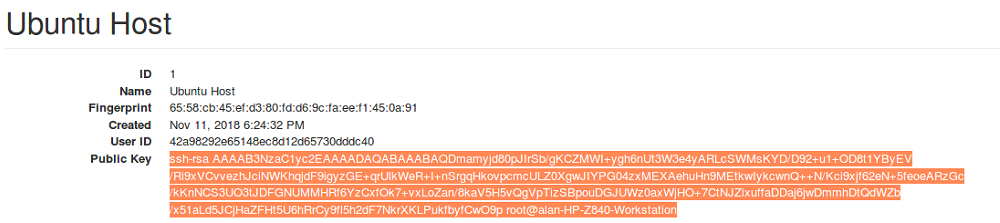
Após realizar as etapas abaixo sem sucesso, não foi possível realizar o login por SSH nas instâncias geradas. Desde já a rubrica solicitada extrapolou em muito o conhecimento necessário para se cumprir a tarefa solicitada, pois o momento em que a mesma foi criada não contou com a modificação e a evolução do Openstack e suas respectivas alterações. Foi solicitado ajuda para solucionar esse problema no Piazza porém não obtive orientação alguma, desde já entrego esse trabalho parcialmente completo mas insatisfeito com a orientação recebida e com o roteiro de aprendizagem desatualizado.

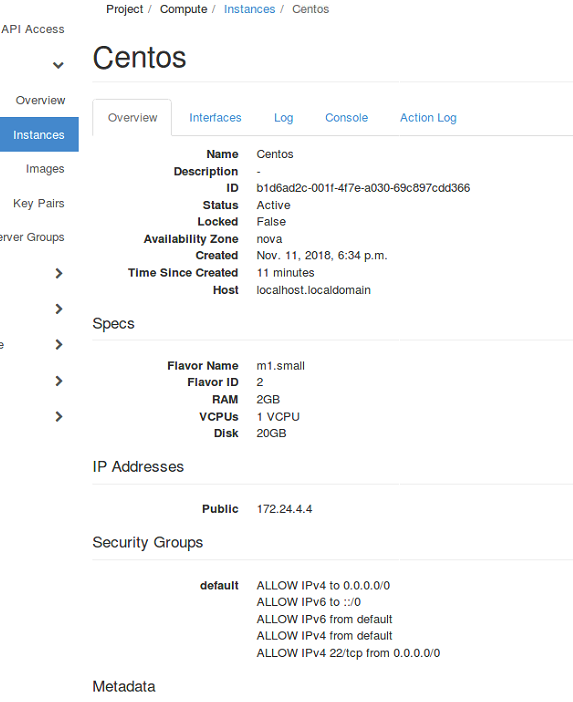


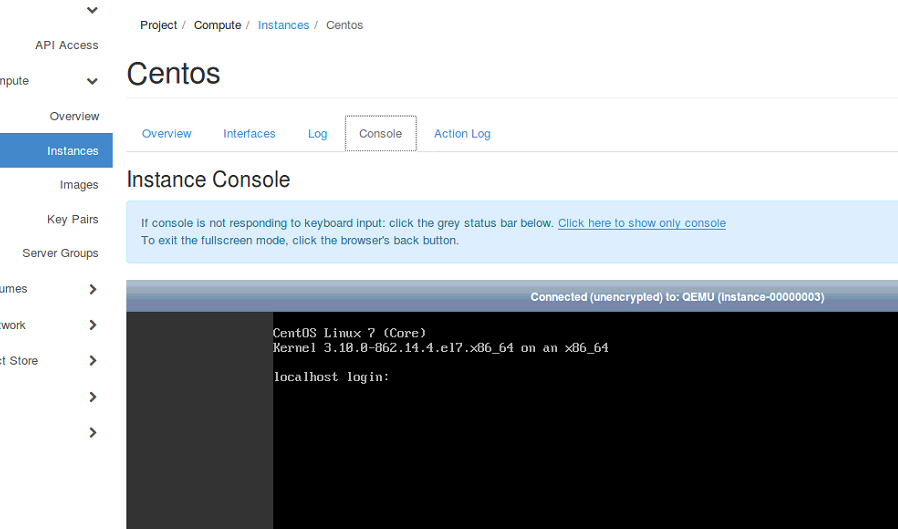


<https://piazza.com/class/jj2wqqf182162y?cid=51>

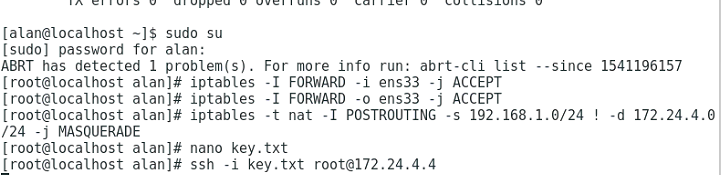
## 2.11 Tentativa final de realizar SSH usando imagem CentOS.

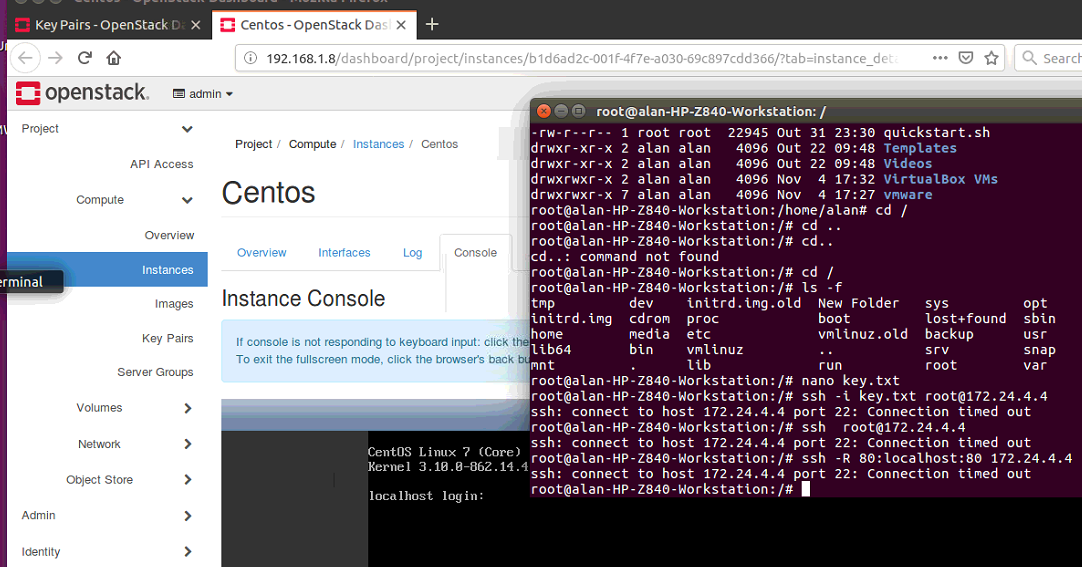












Além dos requisitos apresentados para os capítulos anteriores, suas conclusões devem incluir:

* Se o prazo estabelecido no início do projeto foi adequado para a execução (não há problema se levou mais tempo que o planejado, mas se levou, explique as razões).
* Se os recursos planejados (quantidade de memória, disco, etc.) foram suficientes para colocar a solução em funcionamento. Se não foram, explique as razões e sugira uma nova configuração.
* Se as funcionalidades previstas na solução original funcionaram a contento. Se não foram, explique as razões.
* Quais seriam as melhorias futuras que poderiam ser feitas no projeto executado, adotando soluções adicionais, aperfeiçoando instalações ou tornando o cenário mais complexo.
* Antes da entrega final, todo o conteúdo de seu trabalho (mesmo que sejam apenas documentos do Word e PDFs) deve ser armazenado em um repositório GitHub, cujo link deve estar contido no corpo do trabalho.

# 5. Bibliografia

https://www.redhat.com/pt-br/topics/containers/what-is-docker

https://www.opservices.com.br/o-que-e-docker/

http://sinestec.com.br/blog/o-que-e-container-docker/

https://devops.profitbricks.com/tutorials/install-wordpress-with-docker-on-ubuntu-1604/

https://www.howtoforge.com/tutorial/how-to-install-wordpress-with-docker-on-ubuntu/

https://www.youtube.com/watch?v=0cDj7citEjE