

Reference: Vogel, A.; Griebler, D. *Implantando, Avaliando e Analisando as Ferramentas para Gerenciamento de IaaS OpenStack, OpenNebula e CloudStack*. Laboratory of Advanced Researches on Cloud Computing (LARCC), Technical Report, 2016.

Relatório Técnico de Pesquisa (Atividades de 2015)

Nome do Projeto:

High Performance in Cloud (HiPerfCloud)

Avaliação dos Ambientes de Nuvem IaaS

RT2: Implantando, Avaliando e Analisando as Ferramentas para Gerenciamento de IaaS OpenStack, OpenNebula e CloudStack



ID do Documento:	LARCC-HiPerfCloud-RT2
Versão:	1.2
Autores:	Adriano Vogel, Dalvan Griebler
Objetivo:	Avaliação e análise das ferramentas de gerenciamento de nuvem CloudStack, OpenStack e OpenNebula
Tarefa:	Implantar, avaliar e analisar as ferramentas CloudStack, OpenStack e OpenNebula
Hardware:	3 <i>clusters</i> isolados foram criados usando 4 máquinas idênticas, cada uma com 24 GB de RAM (1333 MHz), processador Intel Xeon X5560 (quad-core 2.80GHz), discos SATA II (7200 RPM) e conectados em uma rede Gigabit (10/1000)
Ambiente:	Sistema Operacional (Ubuntu Server 14.04), Virtualizador (KVM e LXC), OpenStack (vers. Kilo), OpenNebula (vers. 4.8.0), CloudStack (vers. 4.5.2) Benchmarks de Isolamento (Iperf, IOzone, STREAM e LINPACK), Aplicações Paralelas (NPB-MPI e NPB-OMP) e de avaliação de rede (Hpcbench)
Softwares:	GNUPlot (Gráficos), Latex (Documentos)

Tarefa	Responsável	Instituição	Papel	Data
Criado por:	Dalvan Griebler	SETREM	Coordenador	09/01/2016
Editado por:	Adriano Vogel	SETREM	Pesquisador	09/02/2016
Revisado por:	Vera Lúcia Benedetti	SETREM	Colaboradora	09/03/2016
	Fauzi Shubeita	SETREM	Colaborador	09/03/2016
Aprovado por:	Dalvan Griebler	SETREM	Coordenador	15/03/2016
	Ildo Corso	ABASE	Colaborador	15/03/2016
Publicado:	LARCC	SETREM	Laboratório de Pesquisa	20/03/2016

Log de Mudanças do Documento

Versão	Autores	Instituição	Mudança	Data
1	Dalvan Griebler	SETREM	Versão inicial	09/01/2016
1	Adriano Vogel	SETREM	Envio para revisão	09/02/2016
1	Adriano Vogel	SETREM	Envio para aprovação	27/02/2016
1	Adriano Vogel	SETREM	Versão final	15/03/2016
1.2	Dalvan Griebler	SETREM	Atualização no layout e informações	18/02/2017

Lista de colaboradores internos e externos

A baixo é listado (em ordem alfabética) as pessoas que fizeram contribuições para este relatório técnico:

- Adriano Vogel (SETREM)
- Carlos A. F. Maron (SETREM)
- Claudio Schepke (UNIPAMPA)
- Dalvan Griebler (SETREM)

Resumo Geral

O objetivo primário do **Projeto HiPerfCloud** (*High Performance in Cloud*) é avaliação de desempenho em ambientes de nuvens IaaS (*Infrastructure as a Service*) e analisar características de implantação e gerenciamento nas ferramentas disponíveis. Este documento apresenta a continuidade do RT1-2015 [23] e mostra novos resultados em implantações de nuvem privada baseadas em 3 ferramentas: OpenStack, OpenNebula e CloudStack.

Contexto do Relatório

Este documento é o segundo Relatório Técnico *Implantação, Avaliação e Análise das Ferramentas para Gerenciamento de IaaS* relativo ao **Projeto HiPerfCloud** que apresenta resultados de infraestrutura (processador, memória, armazenamento e rede) e aplicações científicas executadas em ambientes de nuvem. Ainda, é apresentada uma análise da robustez das ferramentas e implantações em direção à redundância e alta disponibilidade para máquinas virtuais na nuvem.

Estrutura do Relatório

Este documento inicialmente apresenta a estrutura geral. Posteriormente, computação em nuvem é contextualizada com as ferramentas de gerenciamento OpenStack, CloudStack e OpenNebula, seguida pela análise das características relacionadas ao suporte para flexibilidade e robustez das ferramentas. Ainda, as características dos ambientes implantados, experimentos e resultados são apresentados. Ao final, um estudo em direção de alta disponibilidade e redundância é apresentado e discutido.

Sumário

1	Introdução	1
1.1	Visão Geral	1
1.2	Terminologia	1
1.3	Estrutura deste Documento	2
2	Computação em Nuvem IaaS	3
2.1	Ferramentas <i>Open Source</i> para implantação de nuvens IaaS	4
2.1.1	OpenStack	4
2.1.2	OpenNebula	4
2.1.3	Eucalyptus	5
2.1.4	OpenQRM	5
2.1.5	CloudStack	5
2.1.6	Nimbus	5
2.2	Taxonomia de ferramentas de IaaS	6
2.3	Análise comparativa de ferramentas de IaaS	6
2.3.1	Flexibilidade	7
2.3.2	Resiliência	7
3	Desempenho em Nuvens Privadas IaaS	13
3.1	Desempenho da Infraestrutura e Aplicações Científicas	13
3.1.1	Implantações	13
3.1.2	Metodologia dos testes	15
3.1.3	Experimentos	15
3.1.4	Resultados	15
3.2	Avaliação do Desempenho de Rede em Ambientes de Nuvem	17
3.2.1	Experimentos de Rede	17
3.2.2	Desempenho de Rede	17
4	Um Modelo de Ambiente de Produção na Nuvem	21
4.1	CloudStack	21
4.2	GlusterFS	21
4.3	Em Direção à Alta Disponibilidade	22
4.3.1	Alta Disponibilidade baseada em alocação de VMs em <i>Clusters</i> Homogêneos	22
4.4	Em direção a backup em nuvem privada	22
5	Conclusão	23
A	Instalação das Ferramentas de IaaS	1
A.1	Instalação da ferramenta OpenStack versão Kilo	2
A.2	Instalação da ferramenta OpenNebula versão 4.12	37
A.3	Instalação da ferramenta CloudStack versão 4.5	42
B	Artigos Escritos	53

1. Introdução

Este capítulo apresenta um olhar genérico e introdutório do que será discutido nesse documento, elencando o conteúdo dos capítulos e termos relevantes desse estudo.

1.1 Visão Geral

Neste documento ferramentas de código aberto para a implantação de nuvens do tipo IaaS são analisadas através de uma taxonomia e posteriormente as mais robustas são implantadas, e o desempenho das instâncias oferecidas pelas mesmas é analisado. Além do desempenho de aplicações, tópicos especiais (alta disponibilidade, armazenamento redundante, backup) para implantações de nuvem são discutidos.

1.2 Terminologia

- **Infraestrutura:** Representa os recursos de processamento: Memória RAM, armazenamento, rede e processador.
- **Aplicações Paralelas:** Área da computação de alto desempenho.
- **Benchmark:** Programa para teste específico de determinado recurso ou serviço.
- **Cluster:** Conjunto de computadores interligados por uma rede somando recursos.
- **OpenStack:** Ferramenta de código aberto para gerenciamento de infraestrutura de nuvem IaaS.
- **OpenNebula:** Ferramenta de código aberto para gerenciamento de infraestrutura de nuvem IaaS.
- **CloudStack:** Ferramenta de código aberto para gerenciamento de infraestrutura de nuvem IaaS.
- **Eucalyptus:** Ferramenta de código aberto para gerenciamento de infraestrutura de nuvem IaaS.
- **OpenQRM:** Ferramenta de código aberto para gerenciamento de infraestrutura de nuvem IaaS.
- **Nimbus:** Ferramenta de código aberto para gerenciamento de infraestrutura de nuvem IaaS voltado para ambientes científicos.
- **Iperf:** experimento para avaliação de rede.
- **IOzone:** experimento para avaliação de unidades de armazenamento e sistemas de arquivos.
- **STREAM:** experimento para avaliação da memória RAM.
- **LINPACK:** experimento para avaliação do processador.
- **NPB-MPI:** carga de trabalho composto por diferentes *kernels* de simulação de aplicações paralelas com bibliotecas MPI.

- **NPB-OMP**: carga de trabalho composto por diferentes *kernels* de simulação de aplicações paralelas com bibliotecas OMP
- **Hpcbench**: Experimento para avaliação do desempenho de rede, abrangendo protocolos TCP e UDP e possui diversos experimentos tanto de *throughput* como latência.
- **Taxonomia**: Método que estabelece critérios para classificar em algum determinado assunto.
- **Survey**: Método de pesquisa exploratório e quantitativo que faz um levantamento relacionado com algum tema selecionado.

1.3 Estrutura deste Documento

Este documento está organizado em cinco capítulos:

- Capítulo 1: Apresenta um visão geral deste documento.
- Capítulo 2: Nesta seção, encontra-se o referencial sobre a computação em nuvem e seus serviços, seguido pela apresentação de ferramentas e dos métodos usados para classificar e comparar as ferramentas de código aberto disponíveis.
- Capítulo 3: Apresenta a metodologia utilizada para a execução dos experimentos nos ambientes implantados e os resultados dos testes.
- Capítulo 4: Apresenta e discute sobre as implantações avançadas em direção a alta disponibilidade e redundância para evitar perdas de dados
- Capítulo 5: Conclusão do estudo a partir dos resultados e trabalhos futuros.

2. Computação em Nuvem IaaS

Com a consolidação das redes locais e posteriormente da rede global (internet) usuários individuais e corporações se tornaram capazes de acessarem um serviço computacional disponível em qualquer lugar do mundo. Com isso, o tráfego global de informação vem crescendo exponencialmente. Nos primórdios, recursos computacionais passaram a serem vendidos e disponibilizados através da internet, usando o conceito de computação em grade (*grid computing*), posteriormente já com o surgimento de tecnologias de virtualização, surgiu a computação em nuvem. Este paradigma combinava diversas tecnologias consolidadas (redes de computadores, virtualização, *grid e cluster computing*), e se tornou popular por abstrair a complexidade de configuração da infraestrutura e oferecendo máquinas virtuais sob demanda, que receberam o nome de instâncias. Os serviços de nuvem se popularizaram por oferecerem o pagamento de acordo com a utilização (dados armazenados, percentual de utilização de CPU, memória, tráfego de dados, etc) [3].

A venda de poder computacional para usuários oferecido por terceiros (provedores) passou a se chamar nuvem pública, pois qualquer um com cartão de crédito poderia em questão de minutos ter uma instância acessível, alocada em servidores ao redor do mundo e inacessível fisicamente [33]. O investimento inicial, que sempre foi um grande limitador de infraestrutura de TI por ser caro e sofrer desvalorização e deterioração, a nuvem tornou-se uma alternativa para a redução de custos, mais ágil e flexível provisão de recursos e gerenciamento de serviços de TI. Conseqüentemente, vem ocorrendo um aumento significativo na utilização de serviços da computação em nuvem, por usuários domésticos e setores corporativos, educacionais, governamentais e organizações comunitárias. A melhor utilização dos recursos, economia energética e eficiência pode colaborar para o planeta no conceito de TI verde [5].

Com o passar dos anos e aumento da utilização da nuvem pública diversas preocupações surgiram, principalmente as relacionadas com privacidade, segurança dos dados e desempenho. Armazenar dados corporativos vitais e confiar todo o poder de processamento e armazenamento à terceiros se tornou potencialmente perigoso. Além disso, diversas instituições possuíam hardware e ambientes de TI próprios. Além das preocupações com segurança da informação, diversas pesquisas mostraram preocupações com desempenho nesses ambientes. O uso de virtualização diminui o poder computacional e em alguns casos combinações equivocadas de tecnologias e *drivers* de infraestrutura degradam ainda mais o desempenho. Além disso, em ambientes de nuvem pública o hardware é compartilhado por máquinas virtuais de clientes distintos, cada um com suas cotas, mas mesmo competindo por recursos e uma VM podendo prejudicar o desempenho das demais. Por esses motivos surgiu a chamada nuvem privada, a qual é implantada dentro de uma instituição usando sua infraestrutura e fornecendo recursos computacionais exclusivamente para os usuários locais. Diversas ferramentas de gerenciamento de infraestrutura virtual estão disponíveis, algumas pagas e outras de código aberto e sem custos.

Nuvem privada também se tornou uma alternativa competitiva e largamente utilizada, e surgiu ainda a nuvem híbrida que é uma combinação entre nuvem pública e privada. Dessa forma, frequentemente serviços primordiais e delicados são executados em uma nuvem implantação de nuvem privada enquanto demais serviços secundários são migrados para instâncias de nuvem pública, pois é alternativa barata e flexível. Independente do tipo de nuvem, quando a provisão é de poder computacional (CPU, memória, armazenamento, rede, sistemas operacionais) faz parte do tipo de serviço de nuvem mais popular, IaaS (*Infrastructure as a Service*). Existem outros serviços de nuvem reconhecidos como padrões, PaaS (plataforma como serviço) e SaaS (Software como serviço), as plataformas ou software são executadas na nuvem dentro de um ambiente de IaaS, o que resulta em dependência de recursos e na grande importância dos serviços IaaS.

2.1 Ferramentas *Open Source* para implantação de nuvens IaaS

Atualmente existem diversas ferramentas de código aberto para gerenciamento de infraestrutura virtual de nuvem, algumas dessas soluções são usadas para implantação de nuvens privadas e outras suportam a configuração de um provedor de nuvem pública [35, 14, 38, 39]. Nesse documento foram consideradas todas as ferramentas de código aberto para nuvens IaaS. A praticidade do uso da virtualização que possibilitou a execução de vários sistemas operacionais no mesmo *hardware* se tornou um desafio no ponto de vista do gerenciamento e utilização. A virtualização que é voltada para o *hardware*, para construir um ambiente de nuvem recebeu uma camada de *software* adicional, que é representada pelas ferramentas de IaaS, simplificando e centralizando o gerenciamento de complexas infraestruturas computacionais.

As ferramentas de IaaS são comparadas usando uma taxonomia específica para essas soluções. A análise leva em conta o fato que diversas tecnologias necessárias em camadas diferentes são necessárias para implantação de ambientes de nuvem eficientes e confiáveis. Dessa forma, o comparativo é focado em robustez, que é o resultado da combinação do suporte para flexibilidade e resiliência encontrado nas ferramentas de IaaS. A seguir as ferramentas são apresentadas.

2.1.1 OpenStack

OpenStack é uma ferramenta *open source* altamente conhecida no cenário de IaaS, isso ocorre principalmente por seu projeto ter sido lançado pelas respeitadas Rackspace e NASA, e em seguida diversas gigantes do mundo da tecnologia uniram-se ao projeto. Um dos seus principais objetivos é a interoperabilidade e flexibilidade de implantações de nuvem, possuindo mais de 40 APIs que são componentes independentes que estão disponíveis para instalação. A escolha desses componentes é totalmente customizável de acordo com a demanda de cada arquiteto de nuvem [28].

Alguns componentes se destacam por serem imprescindíveis para uma ambiente de nuvem. Por exemplo, o keystone controla a autorização e autenticação de toda a infraestrutura virtual (demais componentes, serviços, usuários, etc) e precisa ser configurado e “amarrado” aos componentes para a nuvem funcionar [13]. O componente glance que oferece um repositório de imagens para a criação de instâncias na nuvem. A conectividade da nuvem pode ser implementada através do componente nova-network (básico, usando as *bridges linux*) ou usando o neutron (avançado, rede como serviço) [9]. O armazenamento para as instâncias pode ser implementado usando o componente Cinder (armazenamento em blocos LVM) ou usando o robusto swift (armazenamento em objetos). Outro componente relevante é o horizon que oferece interface gráfica (GUI) para usuários e administradores da nuvem. Diversos outros componentes podem ser adicionados a nuvem de forma sistemática [19] e toda a comunicação entre os serviços ocorre usando um mensageiro instantâneo, sendo o RabbitMQ o mais popular.

2.1.2 OpenNebula

OpenNebula surgiu como um projeto de código aberto para virtualização de *data centers* e provisão de serviços de nuvem. Uma característica específica é buscar por uma implantação simplificada e ao mesmo tempo eficiente [26], oferecendo ainda customização para usuários e desenvolvedores.

Essa ferramenta possui uma arquitetura modular, iniciando com os *drivers* básicos que implementar controle sob a virtualização (Libvirt, NFS, dummy) e no núcleo da ferramenta são implementadas as rotinas mais importantes (controle de serviços e usuários, interfaces de rede, monitoramento da infraestrutura, escalonador, transferência entre os nodos, gerencia de

armazenamento, entre outros) [36]. A maioria das tarefas são realizadas através da interação do componente oned com o sistema operacional nativo.

2.1.3 Eucalyptus

Eucalyptus um longo acrônimo de *Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems* [12] é considerado um *bundle* para implantação de ambientes de nuvem, que busca melhorar o gerenciamento da infraestrutura e provisão de serviços. Um aspecto único dessa ferramenta é possuir APIs voltada para integração com Amazon EC2, que diferentemente das demais ferramentas oferecem apenas um suporte para nuvem híbrida.

Eucalyptus possui uma arquitetura voltada para escalabilidade [22]. As principais partes de uma implantação com essa ferramenta são o controlador de nodo, de *cluster* e o de armazenamento. O controlador de nodo [2] é um serviço executado em cada nodo que aloca recursos para as instâncias e monitora a utilização. O controlador de *cluster* é geralmente executado no servidor que gerencia a nuvem, controlando e balanceando as requisições dos clientes. Ainda, o controlador e servidor de armazenamento oferece volumes virtuais para as instâncias e geralmente é um servidor dedicado.

2.1.4 OpenQRM

OpenQRM é uma solução para nuvem dividida em dois produtos, versão comunitário (*open source e grátis*) e *enterprise* (*open source e paga*). Este documento considera para fins de comparação a versão comunitária, que é uma ferramenta de simples instalação e que busca alta disponibilidade através da redundância [27].

[37] cita que OpenQRM possui uma arquitetura modular e suporta módulos de gerenciamento avançados (monitoramento, desempenho, escalabilidade). Os serviços são implementados na forma de *plugins* e possui uma interface gráfica para gerenciamento (administrador de nuvem) e um portal (para clientes).

2.1.5 CloudStack

Cloudstack [6] é um projeto para nuvens privadas e públicas focado em alta disponibilidade e flexibilidade. A infraestrutura possui 2 componentes imprescindíveis, o gerente e o agente de nuvem [32]. Enquanto o gerente controla a infraestrutura virtual e é instalado no servidor principal, o agente é instalado nos nodos escravos que são agrupados em clusters.

Essa solução possui uma interface gráfica completa, onde toda a infraestrutura pode ser montada, o isolamento entre usuários pode ocorrer através de zonas de disponibilidade, grupos de segurança ou através da rede (Vlan, túneis GRE, etc). o armazenamento dos volumes das instâncias é feito na zona primária e precisa de uma zona de armazenamento secundária para armazenar imagens de sistemas operacionais, *templates* e *snapshots*.

2.1.6 Nimbus

Nimbus [25] é uma ferramenta para nuvens IaaS focada em implantações de clusters para o meio científico. Nessas implantações os usuários instanciam máquinas virtuais através de um repositório de imagens (Cumulus) que também controla o armazenamento nos volumes virtuais. Essa ferramenta por ser de código aberto, busca atrair desenvolvedores para flexibilidade e escalabilidade do sistema.

A arquitetura dessa plataforma possui diversas partes [21]. Os clientes interagem com a nuvem através de interfaces de linha de comando (CLI), e podem usar *framework* compatíveis

com a nuvem pública da Amazon. Nimbus possui também um controlador de recursos que usa SSH para se comunicar com os nodos do *cluster* e máquinas virtuais.

2.2 Taxonomia de ferramentas de IaaS

Taxonomia pode ser definida como a ciência da categorização e classificação baseada em métodos pré-definidos e estruturados [20]. O [11] apresenta uma proposta de taxonomia para análise e comparação de ferramentas para implantação de nuvens IaaS. Dessa forma, a taxonomia conceitual foi aplicada nas ferramentas escolhidas para analisar o suporte a flexibilidade nas implantações de cada ferramenta.

A taxonomia [11] é dividida em camadas e as camadas são formadas por itens 2.1. Na base da taxonomia tem-se a camada de abstração de recursos que interage com a virtualização e oferece recursos virtuais (processamento, armazenamento e conectividade) para a nuvem. A camada de núcleo de serviços gerencia os clientes e aloca recursos através de serviços (repositório de imagens, medição e utilização, essa camada conta ainda com o escalonador de recursos da nuvem que controla principalmente a alocação de VMs.

A camada de gerenciamento [30, 31] é responsável por controlar as operações do ambiente de nuvem, principalmente através de grupos de usuários e do gerenciamento de recursos através de interfaces (gráficas, APIs ou CLIs). Outros aspectos do gerenciamento de nuvem que podem ser destacados é o suporte à elasticidade e orquestradores, para alocação em massa de recursos. Já na camada de suporte serviços adicionais são oferecidos, tais como banco de dados, transferências e mensagens entre os servidores, nuvens e redes.

A camada de controle implementa políticas de utilização do ambiente de nuvem. Os acordos de nível de serviço (SLA) garantem critérios e métricas de qualidade e controle, e a medição verifica quanto dos recursos está sendo utilizado. A camada de serviços agregados oferece opções extras como migração de VMs, e soluções para alta disponibilidade e portabilidade. Outra camada vital é a de segurança, que autentica, autoriza e verifica os recursos e usuários da nuvem.

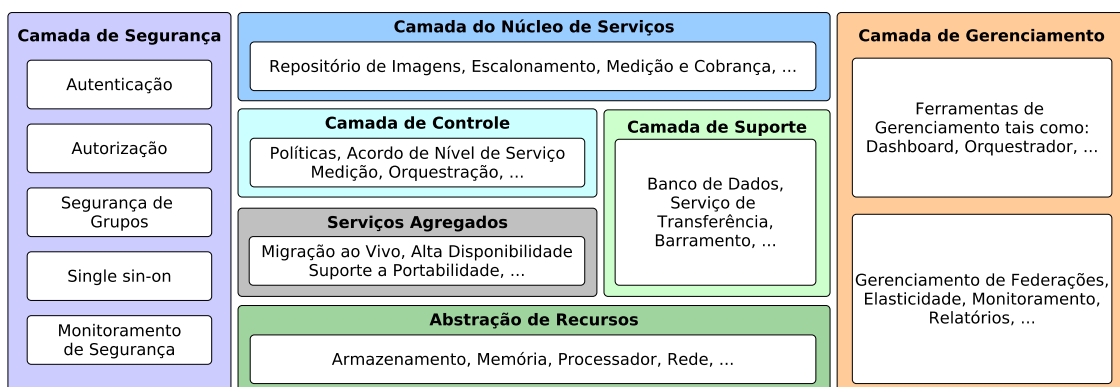


Figura 2.1: Taxonomia conceitual de IaaS. Adaptado de [11]

2.3 Análise comparativa de ferramentas de IaaS

As ferramentas de IaaS possuem objetivos e arquitetura diferentes, por isso se presume que possuam também contrastes no suporte para tecnologias e aplicações. Neste estudo a análise foi voltada para a robustez das ferramentas, que é dividida em flexibilidade e resiliência [34]. Usando métodos específicos, as próximas apresentam uma análise desses aspectos.

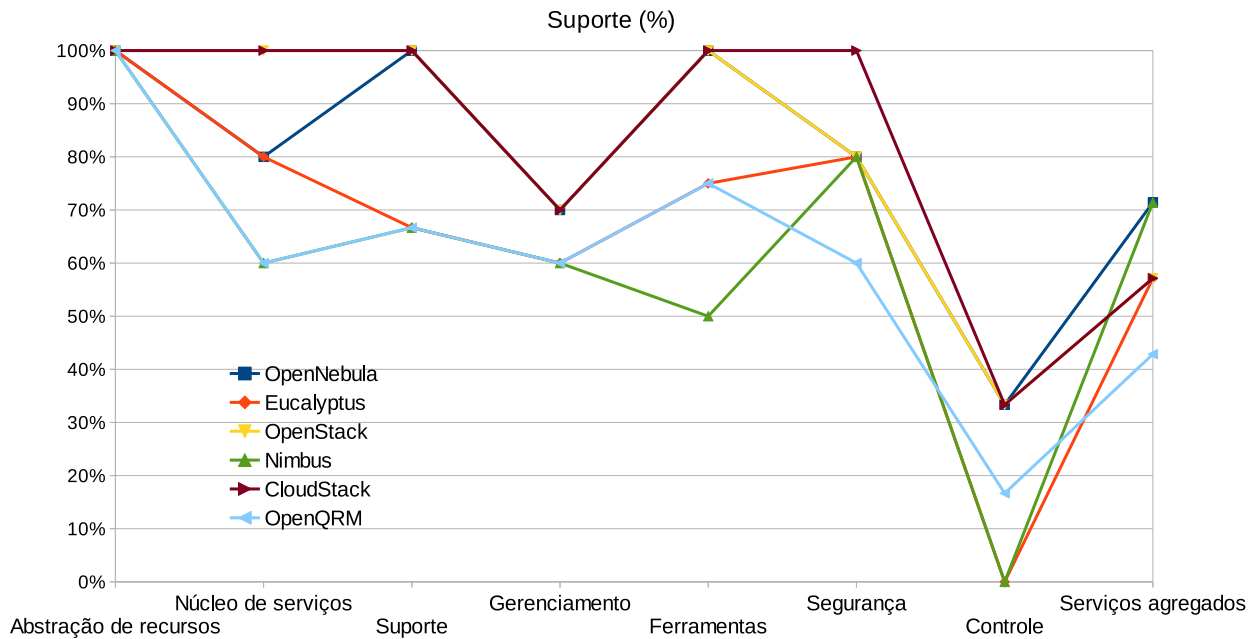


Figura 2.2: Flexibilidade das ferramentas de IaaS(%).

2.3.1 Flexibilidade

A flexibilidade de um sistema computacional é a capacidade de suportar diversos componentes, tecnologias e aplicações [3], esse suporte é relevante para atender as diferentes demandas de usuários e aplicações. As tabelas 2.1 e 2.2 apresentam a análise das ferramentas, o caractere "/" significa que naquele item não foi encontrada referência de suporte para as tecnologias relacionadas, a palavra "interno" significa que existe suporte, porém não está claro qual componente implementa, ou é do núcleo da ferramenta. Outra distinção usada foi a palavra "externo" que significa que o suporte pode ser suprido facilmente a partir da integração de uma tecnologia totalmente compatível.

A Figura 2.2 sumariza os níveis percentuais de flexibilidade de cada ferramenta, que é calculado a partir dos resultados das tabelas 2.1 e 2.2. A taxonomia é dividida em camadas que são formadas por itens, o resultado da figura mostra de acordo com o numero de itens suportados por cada ferramenta em cada camada o relativo percentual (de 0 a 100%), por exemplo, para uma ferramenta atingir 100% em alguma camada precisa ter compatibilidade como todos os tópicos.

Analisando os resultados de flexibilidade, de um modo geral as ferramentas apresentam um suporte pobre nas camadas de controle, serviços e gerenciamento. As ferramentas para implantação de ambientes de nuvem supostamente deveriam aumentar suas preocupações com a flexibilidade para as implantações, pois muitas apresentam um foco comercial forte e acabam deixando de lado o aumento da compatibilidade com as tecnologias com o cenário de constante mudanças. CloudStack teve o nível mais alto de compatibilidade seguido por OpenStack e CloudStack.

2.3.2 Resiliência

O suporte computacional para resiliência é a habilidade de adaptação para mudanças e em caso de falhas manter a disponibilidade dos serviços[4]. Em ambientes de nuvem a possibilidade de manter os serviços disponíveis é fortemente relacionada com as tecnologias de infraestrutura

Suporte	OpenNebula	Eucalyptus	OpenStack
Abstração de recursos			
Computador	Oned	Interno	Nova
Armazenamento	Interno	Walrus	Object storage (Swift) /Block Storage (Cinder)
Volume	Interno	Storage Controller	Nova-Volume
Rede	Virtual Network Manager	Interno	Neutron/Nova-network
Núcleo dos serviços			
Identidade	Interno	IAM API	Keystone
Escalonador	Scheduler	Cluster Controller	Nova-scheduler
Repositório de imagens	Interno	Interno	Glance
Cobrança	/	/	ceilometer
<i>Logging</i>	Interno	Interno	Interno
Suporte			
<i>Bus</i> mensageiro	Interno/RabbitMQ	/	RabbitMQ
Banco de dados	sqlite/MySQL	Sqlite/HSQldb	MySQL/Galera/MariaDB/MongoDB
transferência	Interno	Node Controller	Nova Object store/cinder
Gerenciamento			
Gerenciamento de recursos	Interno	Interno	Nova
Gerenciamento de federação	/	/	/
Gerenciamento de Elasticidade	Auto-scaling	Elastic Load Balancing	Elastic Recheck
Usuários e grupos	Interno	Interno	Interno
SLA	/	/	/
Monitoramento	probe/ssh/OneGate	externo	externo
Relatório	code reporting	/	/
Gerenciamento de incidentes	/	/	externo
Gerenciamento de energia	externo	externo	Blueprint driver
Aluguel	externo	externo	externo
Ferramentas			
CLI	OpenNebula CLI	Euca2ools	OpenStack (CLI)
APIs	Public cloud and Plugins	Public cloud and Plugins	Public cloud and Plugins
<i>Dashboard</i>	Sunstone(Admin UI, User UI)	Admin UI, User UI	Horizon(Admin UI)
Orquestrador	onflow	/	heat
Segurança			
Autenticação	Basic Auth/OpenNebula Auth/x509 Auth/LDAP	LDAP/ AWS IAM	LDAP/Tokens(APIs)/X.509/HTTPD
Autorização	Auth driver	Interno	Keystone
Grupos de segurança	Interno	Interno	Interno
<i>Single sign-on</i>	/	/	/
monitoramento de segurança	externo	externo	externo
Controle			
controle de SLA	/	/	/
monitoramento de SLA	/	/	/
Medição	externo	/	ceilometer
Política de controle	/	/	/
Serviços de notificação	/	/	/
Orquestração	OneFlow	/	heat
Serviços agregados			
Zonas de disponibilidade	Interno	Interno	Interno
Alta disponibilidade	externo	externo	externo
Suporte híbrido	Amazon EC2/Microsoft Azure/IBM	Amazon AWS	HP Helion/Amazon EC2/IBM
<i>Live migration</i>	Interno	Interno	Interno
Portabilidade	/	/	/
Contextualização de imagens	one-context	/	/
Aplicações virtuais	/	/	/

Tabela 2.1: Flexibilidade(a).

Suporte	Nimbus	CloudStack	OpenQRM
Abstração de recursos			
Computador	workspace	Libcloud	interno
Armazenamento	Cumulus	interno	interno
Volume	interno	interno	interno
Rede	workspace Control	interno	interno
Núcleo dos serviços			
Identidade	/	IAM plugin	/
Escalonador	interno	interno	interno
Repositório de imagens	interno	interno	Image Shelf
Cobrança	/	CloudStack Usage	/
<i>Logging</i>	Workspace-service	interno	interno
Suporte			
<i>Bus</i> mensageiro	/	interno/RabbitMQ	/
Banco de dados	interno	MySQL	Postgres/MySQL
Transferência	Workspace-service	interno	interno
Gerenciamento			
Gerenciamento de recursos	interno	interno	interno
Gerenciamento de federação	/	/	/
Gerenciamento de Elasticidade	/	Elastic Load Balancing	/
Usuários e grupos	interno	interno	interno
SLA	/	/	/
Monitoramento	externo	externo	externo
Relatório	/	/	/
Gerenciamento de incidentes	/	interno	interno
Gerenciamento de energia	externo	externo	externo
Aluguel	externo	externo	externo
Ferramentas			
CLI	interno	cloudmonkey	interno
APIs	Public cloud and Plugins	Public cloud and Plugins	Public cloud and Plugins
<i>Dashboard</i>	/	Admin UI	Admin UI
Orquestrador	/	cloudstack Cookbook	/
Segurança			
Autenticação	X.509	SAML/LDAP	LDAP
Autorização	interno	SAML	interno
Grupos de segurança	interno	interno	/
Single sign-on	/	externo	/
Monitoramento de segurança	externo	externo	externo
Controle			
controle de SLA	/	/	/
monitoramento de SLA	/	/	/
Medição	/	/	/
Política de controle	/	/	/
Serviços de notificação	/	interno	interno
Orquestração	/	cloudstack Cookbook	/
Serviços agregados			
Zonas de disponibilidade	EC2 driver	interno	/
Alta disponibilidade	externo	externo	externo
Suporte híbrido	Amazon EC2	Amazon EC2	Amazon AWS
<i>Live migration</i>	interno	interno	interno
Portabilidade	/	/	/
Contextualização de imagens	Nimbus context	/	/
Aplicações virtuais	/	/	/

Tabela 2.2: Flexibilidade(b).

Suporte	OpenNebula	Eucalyptus
Virtualização Completa e Bare Metal	Xen, KVM, Vmware	VMware, KVM, Xen
Tecnologias de armazenamento	NFS, ssh, ceph	ceph, OSG
Formatos de discos	qcow2, vmfs, ceph, lvm, fslvm, raw, dev	qcow2, LVM, raw, vmdk
Rede	dummy, ebttables, VLAN, OVS, vmware	VLAN, bridge, DHCP
SO	Ubuntu, Debian, RedHat, SUSE, CentSO	CentOS, RHEL
Virtualização em Contêiner	Não	Não
Interface gráfica (GUI)	Sim	Sim
Storage de objetos	Sim	Sim
Suporte	OpenStack	Nimbus
Virtualização Completa e Bare Metal	Hyper-V, VMware, Xen, KVM, VirtualBox	Xen, KVM
Tecnologias de armazenamento	LVM, Ceph, Gluster, NFS, ZFS, Sheepdog	local, EC3
Formatos de discos	LVM, qcow2, raw, vhd, vmdk, vdi	qcow2
Rede	Neutron, and B.Switch, Brocade, OVS, NSX, PLUMgrid	DHCP, ebttables
SO	Debian, Ubuntu, RHEL, CentOS, Fedora, Suse	Debian, Ubuntu, RedHat, Gentoo, SUSE
Virtualização em Contêiner	Sim	Não
Interface gráfica (GUI)	Sim	Não
Storage de objetos	Sim	Não
Suporte	CloudStack	OpenQRM
Virtualização Completa e Bare Metal	Hyper-V, Xen, KVM, VMware, VirtualBox	KVM
Tecnologias de armazenamento	NFS, SMB, SolidFire, NetApp, Ceph, LVM	LVM, NFS, GlusterFS
Formatos de discos	LVM, VMDK, VHD, qcow2,	raw, qcow2
Rede	bridge, VLAN, DHCP, DNS, NVP, BigSwitch, OVS	bridge, VLAN, DHCP, DNS, TFTP
SO	Debian, Ubuntu, RHEL, CentOS	Debian, Ubuntu, RHEL, CentOS, OpenSuSE
Virtualização em Contêiner	Sim	Não
Interface gráfica (GUI)	Sim	Sim
Storage de objetos	Sim	Não

Tabela 2.3: Resiliência de ferramentas de IaaS.

envolvidas, essas tecnologias se propriamente configuradas que tornarão a tolerância à falhas possível [18]. Do ponto de vista das ferramentas, elas precisam suportar tecnologias atuais e relevantes para a implantação de ambientes complexos. Na tabela 2.3 as ferramentas são analisadas quanto ao suporte para tecnologias subjacentes de acordo com método de critérios.

O suporte à técnicas de virtualização se refere as tecnologias que são suportadas tanto em virtualização completa ou para-virtualização. Cada virtualizador utiliza meios específicos para emular recursos computacionais e fazer a conversão dos binários entre o hardware e as camadas de software, por isso é relevante o suporte de diversas soluções.

A solução de virtualização usada também impacta nas tecnologias de armazenamento, tipos de redes e formatos de discos suportados, pois cada virtualizador suporta tecnologias específicas. Os sistemas operacionais suportados impactam na possibilidade ou não de se usar virtualização em contêiner, que cria máquinas virtuais que compartilham o mesmo *kernel* do ambiente nativo, o isolamento de recursos ocorre geralmente via usuários e processos. As ferramentas de IaaS necessitam de *drivers* e bibliotecas para interagirem com tecnologias de contêiner. Ainda, o *storage* de objetos é uma forma robusta e inovadora de oferecer armazenamento para usuários, os discos são tratados como objetos e distribuídos de forma escalável pela rede para processamento e tratamento de grande volumes de dados e requisições de aplicações intensivas [8].

A Figura 2.3 apresenta o valor percentual de cada ferramenta nos tópicos. Os resultados da Tabela 2.3 que de acordo com o número de tecnologias suportadas por cada ferramenta em relação ao total de tecnologias apresenta o percentual, por exemplo o 100% ocorre quando todas

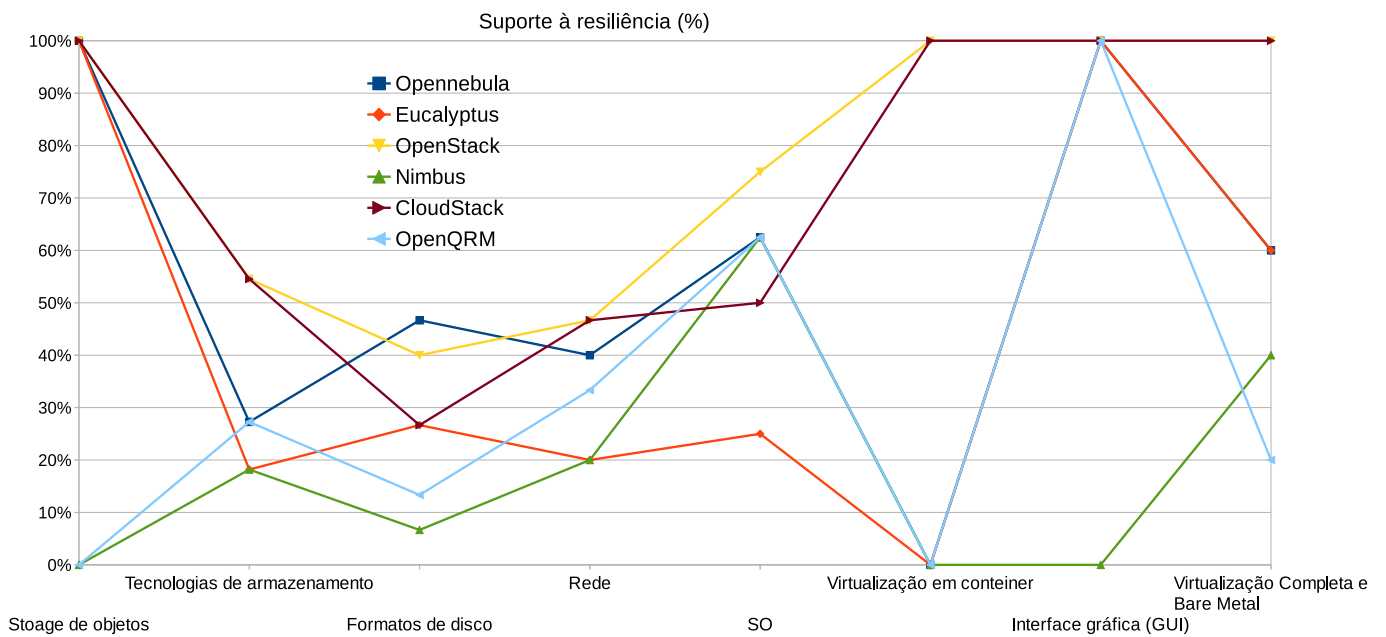


Figura 2.3: Resiliência das ferramentas de IaaS(%).

as tecnologias de algum tópico são suportadas por alguma ferramenta.

A ferramenta OpenStack alcançou os melhores resultados no suporte à resiliência, pois suporta diversas tecnologias de virtualização, armazenamento e redes. CloudStack aparece em seguida, e ambas suportam *storage* de objetos, virtualização em contêiner (usando LXC).

3. Desempenho em Nuvens Privadas IaaS

Como apresentado com Capítulo 2, as ferramentas para implantação de ambientes de nuvem suportam diversas tecnologias e configurações. Nesse seção, ambientes de nuvem usando ferramentas distintas foram implantados e o objetivo principal é comparar os resultados e apresentar um viés inovador nos resultados. Na Seção 3.1 são apresentados resultados da infraestrutura e aplicações científicas, e na Seção 3.2 são apresentados os resultados de diversos experimentos de rede (vazão e latência) nos ambientes de nuvem.

3.1 Desempenho da Infraestrutura e Aplicações Científicas

3.1.1 Implantações

Para a execução de experimentos, três *clusters* isolados foram criados usando 4 máquinas idênticas, cada uma com 24 GB de RAM (1333 MHz), processador Intel Xeon X5560 (quad-core 2.80GHz), discos SATA II (7200 RPM) e conectados em uma rede Gigabit (10/1000). No ambiente nativo e nas instâncias foi utilizado o sistema operacional Ubuntu *Server* 14.04 (kernel 3.19.0). Optou-se por implementar os ambientes usando virtualização completa através do KVM (versão 2.0.0) e as ferramentas de nuvem OpenStack versão Kilo, OpenNebula 4.12 e CloudStack 4.5.2. Como foi utilizado 4 lâminas independentes nas 3 implantações distintas, os discos foram particionados usando *multiboot*.

A arquitetura das implantações foi similar. Usando as diferentes ferramentas de IaaS, cada ambiente teve um servidor agindo como gerente de nuvem, o qual controla a infraestrutura (recursos e usuários) e aloca as instâncias nos nodos. Controlados pelo gerente de nuvem, o virtualizador e agente de nuvem foram instalados em cada servidor (nodo) que ofereciam os recursos virtuais para as instâncias de nuvem. Ambas as implantações utilizaram os volumes distribuídos pela rede, OpenNebula e CloudStack usando o protocolo NFS (Network File System) e imagens QCOW e OpenStack usou o protocolo iSCSI (Internet Small Computer System Interface) e imagens LVM. Em ambas as implantações a conectividade entre as instâncias foi usando as *Bridges Linux* e o *driver* VirtIO. Para aumentar o poder de processamento, em cada servidor foi alocada apenas uma instância que tinha todos os recursos disponíveis.

Implantação da Ferramenta OpenStack

Para criação de instâncias virtuais e execução de experimentos com a ferramenta OpenStack, foi configurado um servidor como controlador de nuvem (armazenamento, rede, serviços, etc) e os nodos de nuvem nos quais as instâncias são alocadas e executadas. Nessa implantação os seguintes componentes do OpenStack e serviços foram instalados no controlador de nuvem:

- Servidor NTP- Para manter sincronizado os horários com os nodos da nuvem.
- Banco de dados - MySQL para armazenar informações da nuvem.
- Servidor RabbitMQ - Para coordenar operações e comunicação entre os serviços da nuvem.
- Keystone - Serviço de identidade que gerencia a autenticação de toda a nuvem.
- Glance - Servidor de imagens e *templates* de SO para os usuários da nuvem

- Nova - Gerenciamento de recursos
- Cinder - Oferece armazenamento em blocos para as VMs.
- Nova-Network - Implementa as redes virtuais para as instâncias
- Horizon - Oferece uma interface gráfica para gerenciamento da infraestrutura.

Nos nodos da nuvem foi instalado os pacotes de rede e configuradas as *bridges* e o componente *nova-compute* que interage com o nodo controlador da nuvem e com o *Libvirt* para execução dos serviços da nuvem. Detalhes adicionais da instalação, configuração e utilização da nuvem está disponibilizado como apêndice desse documento.

Implantação da Ferramenta OpenNebula

O ambiente de testes OpenNebula foi instalado sob o sistema operacional nativo Ubuntu 14.04.3 amd64 e na versão do *kernel* Linux 3.19.0. As especificações dos servidores se encontra na seção 3.1.1 e foi usada uma partição de 600 GB configurada a partir de um RAID 5 (exemplo `/dev/md0p1`), tanto nos *frontend* quanto nos nodos da nuvem.

Um dos primeiros passos para a instalação do OpenNebula é atualizar o sistema e configurar as *bridges* Linux e o endereço IP de cada servidor. A versão instalado do OpenNebula foi a 4.12.0, por era a mais recente no momento da instalação. Nesta implantação, adotou-se o padrão recomendado pelo manual de instalação do OpenNebula no Ubuntu *Server* 14.04, utilizando o virtualizador KVM.

Os repositórios oficiais do OpenNebula para a versão foram adicionados para a *sources list* do sistema operacional. O servidor principal que gerencia a nuvem OpenNebula é chamado de *frontend*, a instalação pode ser feita pelos repositórios, que baixa e configura as bibliotecas, pacotes e *drivers* necessários. Entre os principais está o pacote OpenNebula, o OpenNebula-sunstone que é a interface web de gerenciamento e o servidor NFS que é o modo convencional de fornecer para o *cluster* virtual.

Nos nodos da nuvem, é instalado o pacote *opennebula-node* que comunica com o *Frontend* e também onde o cliente dos diretórios é montado. A interação com a nuvem pode ser através do *Sunstone* ou pelas interfaces de linha de comando (CLI). Diversas customização e configurações avançadas podem ser feitas, seguindo a documentação oficial [29].

Implantação da Ferramenta CloudStack

Para comparar o desempenho entre as nuvens distintas as implantações são semelhantes, a instalação com a ferramenta CloudStack seguiu o modelo das demais ferramentas, com um servidor de controla a nuvem e os nodos da nuvem onde as VMs são alocadas. Na ferramenta CloudStack isso ocorre através do componente *cloudstack-management* e *cloudstack-agent* respectivamente.

A instalação do servidor que controla a nuvem foi iniciada com os pacotes básicos do sistema operacional, instalação do servidor NTP e do pacote *bridge-utils* (para implementar as interfaces de rede *bridges*). Após foi instalado o banco de dados MySQL para controlar as entradas do ambiente e os papéis da nuvem com o banco de dados. Em seguida é instalado o componente *cloudstack-management* que baixa todos os pacotes necessários para o gerenciamento da nuvem.

O nodo gerente da nuvem também foi usado para oferece armazenamento às instancias, nessa implantação foi instalado o servidor NFS e pelos menos dois diretórios precisam ser exportados (para a zona primária e secundária de armazenamento).

Nos nodos da nuvem é imprescindível a instalação do componente *cloudstack-agent* que baixa todos os pacotes necessários para a nuvem (EX: Libvirt, KVM, QEMU). Em cada nodo os diretórios NFS exportados no servidor precisam ser montados para serem acessíveis às VMs

executadas na nuvem. A interface gráfica pode ser acessada pelo servidor gerente da nuvem, e requer algumas configurações iniciais (Criação de redes, *clusters*, zonas, armazenamento).

3.1.2 Metodologia dos testes

A metodologia usada se baseou na forma com que a literatura lida com os mesmo experimentos computacionais, o renomados *micro-benchmarks* escolhidos foram executados 40 vezes e os resultados apresentam a média dos resultados e a comparação entre os ambientes de nuvem.

3.1.3 Experimentos

Os experimentos executados para analisar o desempenho da infraestrutura foram os seguintes:

- **LINPACK**: esse experimento executa tarefas de uso exaustivo de CPU através de cálculos para resolução de matrizes[10], que foram do tamanho de 8000x8000.
- **STREAM**: nesse teste a banda de memória é medida usando vetores de kernel (*Add, Copy, Scale, Triad*) [24], usando blocos de dados maiores que o sistema de *cache*.
- **IOzone**: esse experimento avalia o desempenho de I/O do sistema de arquivos através de quatro operações, escrita, leitura, reescrita e releitura [16], as taxas de transferência foram analisadas usando um arquivo do tamanho de 5 GB.
- **IPerf**: esse teste faz uma medição básica de transferências de rede [17].

O desempenho de aplicações científicas foi avaliado nas três implantações de nuvem usando o renomado NAS [1] com os seguintes *kernels*:

- **Embarrassingly Parallel (EP)**: essa aplicação é caracterizada por independência de cada processo.
- **Multi Grid (MG)**: faz uso exaustivo de memória e comunicação através de operações sequenciais.
- **Integer Sort (IS)**: faz uso de memória de forma randômica com operações de ordenação.
- **Fourier Transform (FT)**: essa aplicação faz uso de comunicação para as operação de *Fourier*.

3.1.4 Resultados

Com a implantação dos ambientes planejados e criação das instâncias de nuvem foi possível a execução dos experimentos escolhidos na metodologia.

Desempenho da Infraestrutura

Os experimentos de infraestrutura foram executados em quatro ambientes distintos e isolados: ambiente nativo e instâncias de nuvem das ferramentas OpenNebula, OpenStack e CloudStack. A Figura 3.1 apresenta os resultados dos experimentos nos ambientes implantados.

De modo geral os resultados demonstraram perdas pequenas de desempenho comparando o ambiente nativo com o virtualizado, um exemplo são os resultados de processamento com o Linpack que foram próximos entre os ambientes. Por outro lado, os resultados de desempenho do armazenamento com o experimento IOzone tiveram mais variação, o que mostra a variação

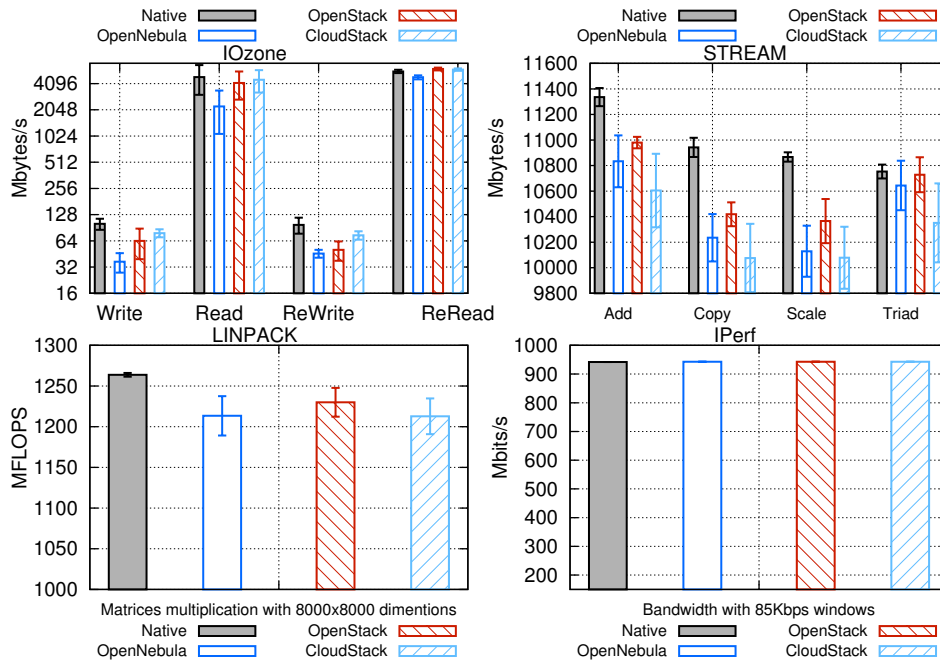


Figura 3.1: Desempenho da Infraestrutura.

devido a carga de trabalho. Os resultados de discos com a ferramenta OpenNebula foram piores que os demais ambientes, e nas operações de leitura e Releitura os resultados não foram muito diferentes, os contrastes ficaram nas operações de escrita e reescrita.

Os experimentos com memória foram um pouco melhores nas instâncias da ferramentas em comparação com OpenNebula e CloudStack. Por outro lado, o desempenho de rede foi muito próximo entre todos os ambientes.

Desempenho de Aplicações Científicas

A Figura 3.2(A) mostra os resultados de desempenho de alguns *kernels* OMP, esses experimentos também foram executados em uma máquina do ambiente nativo e instâncias das ferramentas OpenStack, OpenNebula e CloudStack. Os resultados foram quase os mesmos nos diferentes ambientes, sem diferenças significantes.

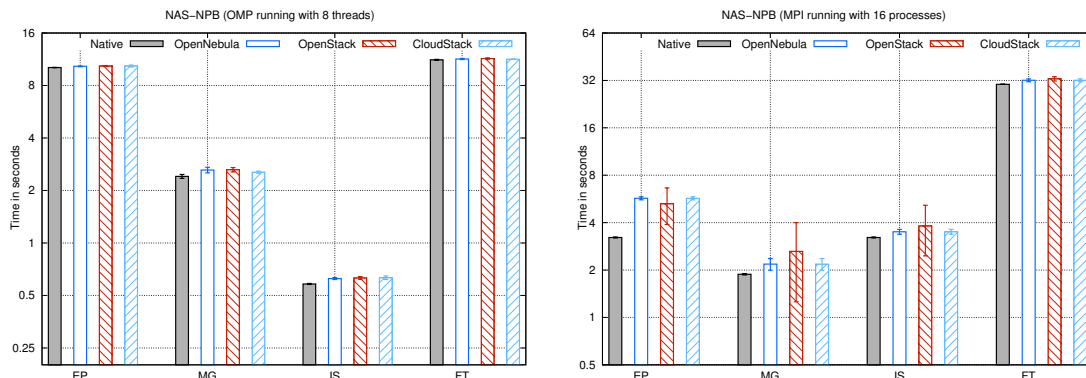


Figura 3.2: Desempenho NAS.

Também foram executados *kernels* do NAS-MPI nos ambientes implantados, esses experimentos foram executados repetidamente entre duas máquinas usando todos os recursos de processamento (até 16 processos) e o mesmo nos ambientes de nuvem, entre duas instâncias de

cada ferramenta. A Figura 3.2(B) apresenta os resultados dos *kernels* NAS, o ambiente nativo teve desempenho melhor que as máquinas virtuais. As instâncias executadas na ferramenta OpenStack tiveram um desvio padrão o que prejudicou o desempenho, enquanto CloudStack e OpenNebula tiveram um desempenho levemente melhor. Porém, as diferenças muito pequenas, o que demonstra que existem perdas de desempenho em ambientes virtualizados que usam memória distribuída enquanto com memória compartilhada o desempenho sofre um impacto menor.

3.2 Avaliação do Desempenho de Rede em Ambientes de Nuvem

Um aspecto relevante que tem um papel essencial para a computação em nuvem são as redes de computadores. Isso ocorre porque atualmente o processamento é distribuído colaborativamente entre nodos de *clusters*, sem conectividade isso seria impossível. Além disso, o diferencial da computação em nuvem é oferecer os recursos computacionais como serviços, o que só é possível graças as redes de computadores. Consequentemente, o desempenho e disponibilidade de aspectos de rede são muito importantes para garantir qualidade dos serviços e isso requer análises e entendimento.

3.2.1 Experimentos de Rede

As implantações dos ambientes nuvem foram semelhantes ao apresentado na Seção 3.1, a principal diferença é que os experimentos de rede foram também executados usando a virtualização em contêiner com LXC. Isso é possível através da integração entre a ferramenta de IaaS CloudStack e com servidores usando LXC para hospedar instâncias. Os experimentos foram executados ainda no ambiente nativo e com o virtualizador KVM com as ferramentas OpenStack, CloudStack OpenNebula. O *benchmark* Hpcbench [15] foi escolhido para execução com os seguintes experimentos:

- **Throughput Exponencial:** Esse experimento inicia com o tamanho de mensagem de 1 Byte e o tamanho do pacote é aumentado exponencialmente. Tamanhos diferentes impactam no *throughput* devido ao número de transferência necessárias, com tamanhos de mensagens maiores o *throughput* deve aumentar também.
- **Throughput Unidirecional:** Esse é o teste mais simples e mais usado em redes, sendo um nodo envia e outro recebe os pacotes.
- **Throughput Bidirecional:** Nesse experimento, dois nodos (cliente e servidor) enviam e recebem pacotes simultaneamente.
- **Latência:** Esse experimento considera o tempo de transferência de um pacote, quando é enviado, o tempo que demora até ele retornar.

3.2.2 Desempenho de Rede

A execução dos experimentos foi entre duas máquinas de cada ambiente implantado, e os resultados são apresentados relacionados com as transferências entre esses dois nodos, usando protocolos TCP e UDP. A Figura 3.3 apresenta os resultados nos cinco ambientes, fica evidente que tamanhos de pacotes pequenos degradam o *throughput*, que só aumenta com pacotes maiores. O ambiente nativo teve os melhores resultados seguido pelo ambiente CloudStack-LXC e pelas instâncias usando o KVM.

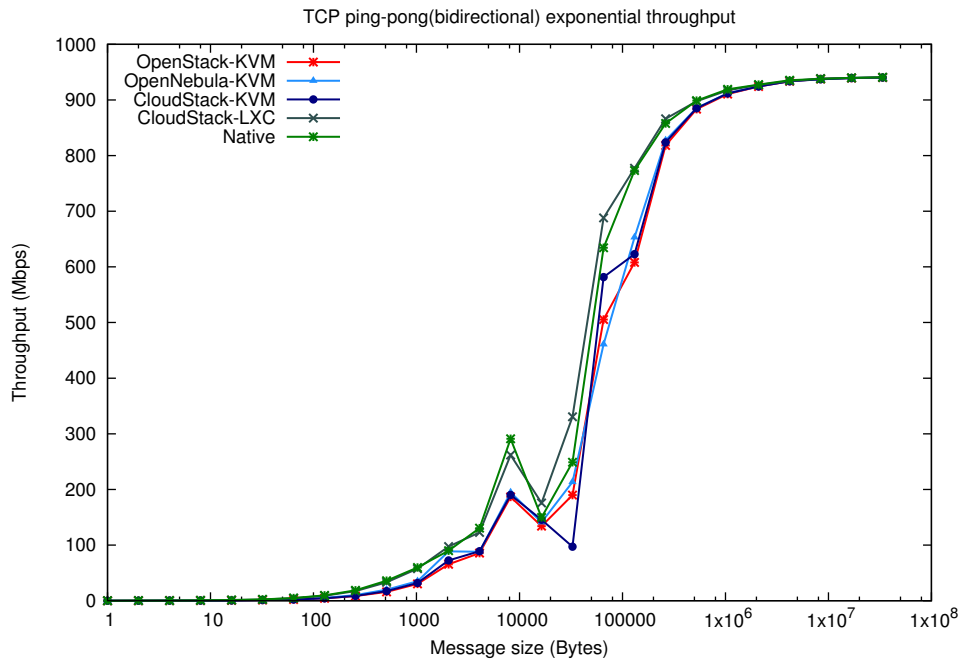


Figura 3.3: *Throughput* TCP Exponencial.

Na Figura 3.4 os resultados de *throughput* nos ambientes usando os protocolos TCP e UDP são mostrados. Comparando entre os ambientes, os resultados foram muito próximos, e o protocolo UDP foi um pouco melhor devido a não monitorar a transferência, focando no envio.

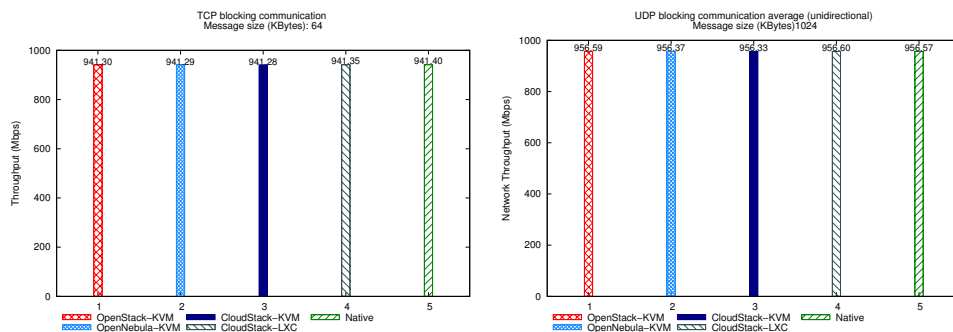


Figura 3.4: TCP e UDP *Throughput* Unidirecional.

A Figura 3.5 mostra os resultados customizando parâmetros dos pacotes. Os tamanhos de mensagem dos pacotes, quantidade de dados que pode ser transferida (MSS) e tamanho do *buffer* foram customizados para ver o desempenho dos ambientes quando as configurações de rede não são as usuais. Os resultados tiveram uma variação maior, o que diminui os contrastes (ganhos ou perdas) comparando os ambientes.

A Figura 3.6 apresenta os resultados de testes bidirecionais, quando os dois nodos enviam e recebem pacotes simultaneamente. Esse experimento apresentou resultados pobres usando o protocolo TCP devido ao controle de tráfego e verificação implementado pelo protocolo e também sofre influência do nível de prioridade do experimento nas filas do escalonador do sistema operacional. Usando o protocolo UDP, os resultados de desempenho foram significativamente melhores, a velocidade foi dobrada se comparada com a transferência unidirecional (Figura 3.4). A comparação entre as instâncias dos ambientes distintos mostra que a ordem se manteve, sendo o nativo o melhor, seguido pelo ambiente baseado em LXC e após, as VMs executadas usando o virtualizador KVM.

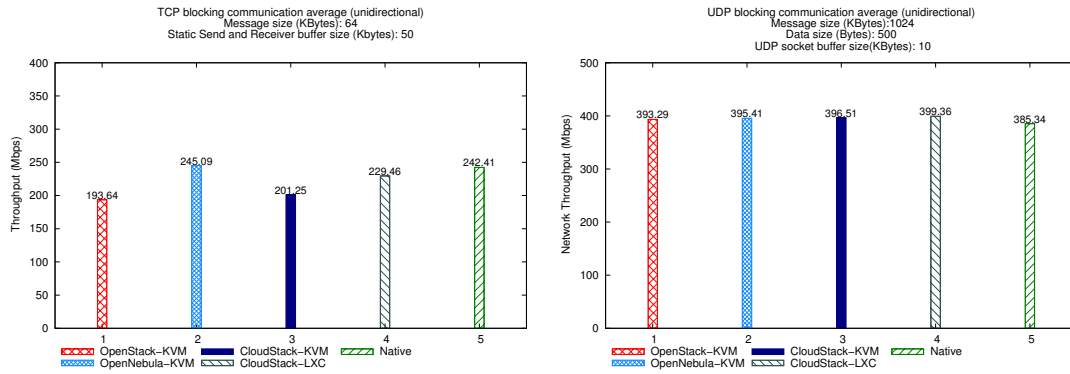


Figura 3.5: TCP e UDP *Throughput* Usando *Buffer* e tamanhos MSS customizados.

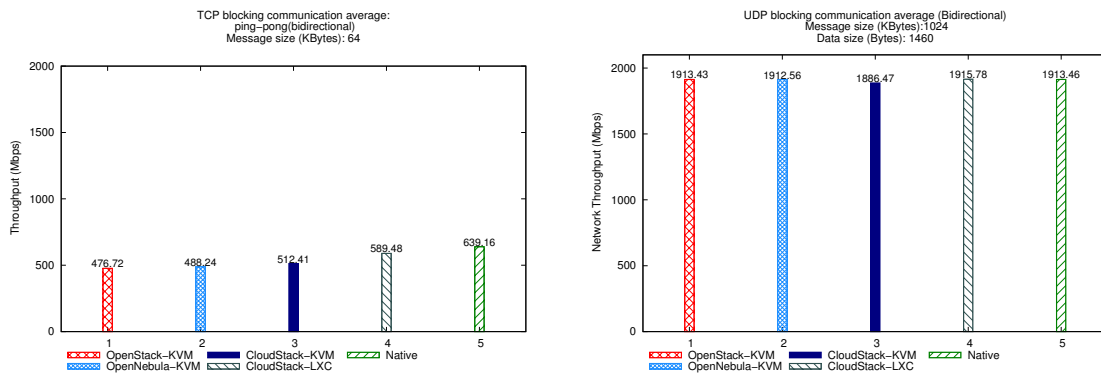


Figura 3.6: TCP e UDP Bidirecional *Throughput*.

Na Figura 3.7 são apresentados os resultados de latência TCP e UDP. Foi usado o RTT (*Round Trip Time*) para medir a latência, considerando o intervalo em micro-segundos entre uma instância iniciar uma conexão com outra, até receber o datagrama de resposta. Dessa forma, quanto menor o tempo, melhor é o ambiente. Contrastando com os resultados anteriores relacionados com o *throughput* de rede, na latência foram encontrados resultados significativamente diferentes entre os ambientes. As instâncias de nuvem usando as três ferramentas e o mesmo virtualizador KVM precisaram do dobro de tempo para completarem as operações, quando comparando com o ambiente nativo e o baseado em LXC. O ambiente nativo foi novamente o melhor, e o ambiente LXC usando as mesmas interfaces *bridges* de rede foi melhor que os ambientes que usam o KVM.

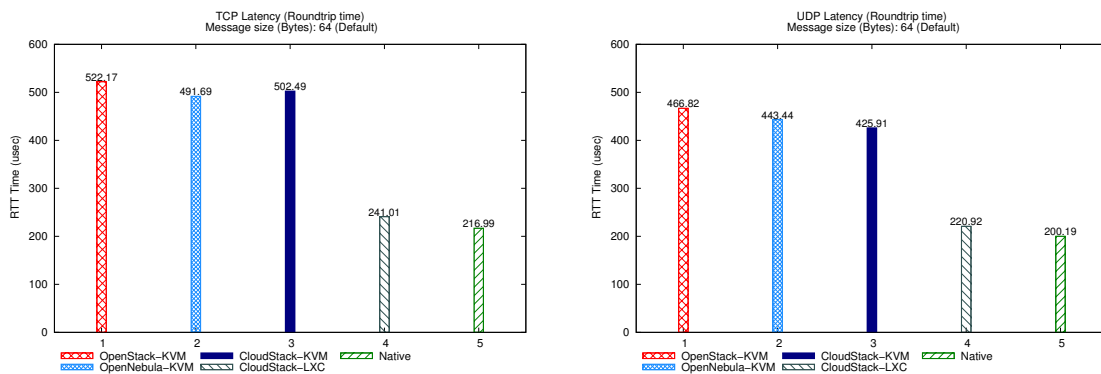


Figura 3.7: Latência.

4. Um Modelo de Ambiente de Produção na Nuvem

Os capítulos anteriores apresentaram análises de ferramentas de nuvem relacionados com robustez e desempenho. Esses resultados são relevantes para qualquer ambiente real de nuvem, pois a flexibilidade e resiliência das ferramentas impacta diretamente no ambiente, pois o suporte para integração com tecnologias e provisão de recursos para usuários define o quão eficiente é um ambiente de nuvem. O mesmo ocorre com o desempenho, o poder de processamento e I/O resulta na qualidade dos serviços oferecidos e executados na nuvem.

Além das implantações de nuvem que foram rodados os experimentos do capítulo anterior, existe uma enorme demanda por tolerância a falhas, redundância e backup e, conseqüentemente por alta disponibilidade. Ambientes de nuvem, pela flexibilidade e escalabilidade são recomendados para essas necessidades. Portanto, um ambiente de nuvem de produção foi implantado em direção de alta disponibilidade dos serviços na nuvem.

De acordo com os requisitos, o protótipo do ambiente foi planejado através da ferramenta de nuvem CloudStack devido a sua arquitetura flexível e voltada para alta disponibilidade. Pela confiabilidade e bom desempenho, o virtualizador KVM e formato de imagens QCOW foram escolhidos. Essas tecnologias raízes e demais secundárias foram selecionadas e escolhidas planejando uma implantação eficiente e tolerante à falhas. Outro aspecto muito importante para o ambiente de nuvem, por isso nesse foi escolhida GlusterFS integrada na nuvem e com volumes replicados para evitar perdas de dados. As próximas seções detalham como cada tecnologia e foi usada e integrada com o ambiente como um todo.

4.1 CloudStack

A ferramenta de IaaS CloudStack foi instalada seguindo a documentação oficial [7] usando o sistema operacional Ubuntu Server 14.04. Um servidor do ambiente foi configurado como gerente de nuvem, ou seja através do pacote *cloudstack-management*, esse nodo controla a utilização dos recursos, usuários e aloca a VMs nos nodos clientes agrupados em *clusters*. O agrupamento dos nodos numa nuvem CloudStack só é possível através da homogeneidade do ambiente, ou seja, todo o *hardware* e *softwares* precisam ser os mesmos. Um dos benefícios da utilização de clusters é a possibilidade de alta disponibilidade e balanceamento de carga.

Outro requisito para alta disponibilidade de VMs em uma nuvem CloudStack é armazenamento distribuído, ou seja, os volumes e ISOs usadas pelas máquinas virtuais precisam estarem acessíveis para todos os nodos do cluster. Com a infraestrutura instalada apropriadamente, as instâncias podem ser criadas com a opção "HA"habilitada.

4.2 GlusterFS

A tecnologia de armazenamento GlusterFS foi integrada com a nuvem CloudStack para oferecer volumes virtuais as instâncias virtuais. O lado do servidor do GlusterFS foi instalado em duas máquinas rodando o sistema operacional Ubuntu Server 14.04, dessa forma, ambas tiveram os pacotes do servidor instalados e posteriormente um volume replicado foi criado e sincronizado entre os servidores. O lado cliente é a ferramenta CCloudStack onde esse volume foi adicionado como zona primária de armazenamento (armazenamento dos discos das máquinas virtuais), e as informações ficam duplicadas em cada servidor para assim evitar perda de dados.

4.3 Em Direção à Alta Disponibilidade

Nas seções anteriores as principais tecnologias usadas na implantação do modelo de ambiente de produção foram apresentadas e foi detalhado como foram configuradas. Alta disponibilidade é algo complexo que depende de vários níveis de serviços e paridades. Iniciando no nível do *hardware*, falha de algum componentes pode resultar e indisponibilidade, a energia que é usada para manter os equipamentos ligados não pode falhar também. Nos nível está os sistemas operacionais e virtualizadores, que oferecem os recursos para os serviços da nuvem.

No caso de alguma falha nas camadas inferiores, as ferramentas para gerenciamento de infraestrutura de nuvem estão avançando para controlarem a disponibilidade, isso geralmente ocorre através da redundância dos servidores.

4.3.1 Alta Disponibilidade baseada em alocação de VMs em *Clusters* Homogêneos

A ferramenta CloudStack oferece uma forma de alta disponibilidade para as máquinas virtuais, através da alocações em servidores agrupados em *cluster*. Cada VM é alocado em um determinado servidor, e a ferramenta monitora a disponibilidade do servidor e em caso de alguma falha seja detectada a máquina é automaticamente transferida para outro servidor do mesmo *cluster*. Para isso, geralmente a VM precisa ser reiniciada, mas em questão de minutos os serviços executados retornam. É possível ainda oferecer redundância e sincronização entre as máquinas virtuais, para no caso de falha de alguma, outra continua a execução de determinada aplicação, exemplos de tecnologias são Heartbeat, Corosync, Keepalived e Pacemaker.

4.4 Em direção a backup em nuvem privada

Além da necessidade pela disponibilidade e confiabilidade em ambientes de produção, existe uma grande necessidade por integridade de dados virtuais. Uma técnica muito usada para evitar perdas de dados é realizar backup dos dados, sempre mantendo pelo menos uma cópia. Diversos provedores já oferecem o recurso de backup em nuvem, que pode ser contratado e pago de acordo com a utilização. Existe também a alternativa da instalação de instalar servidores próprios para backup, onde uma determinada tecnologia é instalada para oferecer esse serviço exclusivamente para um domínio.

Existem diversas soluções para backup, e nesse estudo consideramos as alternativas gratuitas e de código aberto. Nesse segmento, a solução mais popular é o bacula. Analisando essa tecnologia ficou evidente que é uma solução robusta e que atende várias demandas. Consequentemente, essa solução é mais complexa para implantar e gerenciar, além do fato que necessita para o correto funcionamento uma infraestrutura robusta e complexa. Por isso, foi testada uma BackupPC que é mais simplista e facilmente atende aos objetivos de um ambiente de nuvem privada (Backup de máquinas virtuais). Essa solução foi instalada so sistema operacional *Ubuntu Server* e foi habilitada também a interface gráfica para gerenciamento via *Web*, que possibilita a configuração dos clientes de backup, os agendamentos das tarefas de backup e restauração.

5. Conclusão

Os benefícios para a migração dos serviços computacionais para serviços de nuvem são evidentes. Os principais desafios são relacionados com desempenho, gerenciamento e segurança nesses ambientes. Enquanto os provedores de nuvem pública oferecem recursos computacionais sob demanda e pagos conforme a utilização, ferramentas de gerenciamento de infraestrutura surgiram oferecendo a alternativa de se implantar nuvens privadas, com níveis adicionais de controle, privacidade e customização. Essas ferramentas possuem profundos contrastes tanto na arquitetura, implantação, tecnologias e serviços suportados.

Por isso, as ferramentas de código aberto disponíveis para a implantação de ambiente de nuvem foram analisadas e comparadas quanto ao suporte para flexibilidade e resiliência em implantações. Os resultados mostraram a ferramenta CloudStack como a mais flexível e OpenStack mais resiliente. A combinação de flexibilidade e resiliência resulta em robustez, o que reflete no nível de compatibilidade e customização possível para um ambiente de nuvem, isso aliado ao fato de cada aplicação ter características e requisitos diferentes faz com que a robustez seja extremamente importante para QoS.

Além da avaliação das ferramentas, o desempenho das implantações usando o virtualizador KVM mostrou resultados próximos no desempenho da infraestrutura e de aplicações científicas. Em testes específicos de rede ficou evidente que o *throughput* entre ambientes de nuvem KVM, LXC e nativo são próximos. Por outro lado, a latência nas implantações de nuvem usando KVM foram altas em comparação com os ambientes nativos e LXC.

Uma análise e implantação de ambiente de nuvem em direção a alta disponibilidade para serviços na nuvem foi apresentada usando a ferramenta CloudStack. Nessa implantação, os nós são agrupados em *clusters* que formam zonas de disponibilidade, no caso de falha em algum dos nós, as máquinas virtuais são transferidas e reiniciadas em outro servidor. Esse nível de disponibilidade e tolerância a falhas atingido no nível de infraestrutura se combinado com redundância no nível das aplicações pode atingir facilmente elevados níveis de confiabilidade.

Referências Bibliográficas

- [1] Bailey, D.H., Barszcz, E., Barton, J.T., Browning, D.S., Carter, R.L., Dagum, L., Fatoohi, R.A., Frederickson, P.O., Lasinski, T.A., Schreiber, R.S., et al.: The nas parallel benchmarks. *International Journal of High Performance Computing Applications* 5(3), 63–73 (1991)
- [2] Baun, C., Kunze, M., Nimis, J., Tai, S.: *Cloud Computing: Web-Based Dynamic IT Services*. Springer Berlin Heidelberg (2011), <https://books.google.com.br/books?id=b1skkacCvoC>
- [3] Buyya, R., Vecchiola, C., Selvi, S.: *Mastering Cloud Computing*. McGraw Hill (2013), <https://books.google.com.br/books?id=VSDZAgAAQBAJ>
- [4] Castano, V., Schagaev, I.: *Resilient Computer System Design*. Springer International Publishing (2015), https://books.google.com.br/books?id=4_RICAAAQBAJ
- [5] Chandrasekaran, K.: *Essentials of Cloud Computing*. Taylor & Francis (2014), <https://books.google.com.br/books?id=-GhYBQAAQBAJ>
- [6] Cloudstack: *Cloudstack roadmap* <<https://cloudstack.apache.org/>> (2015), last access Dec, 2015
- [7] CloudStack, A.: *CloudStack Installation*, <<https://docs.cloudstack.apache.org/projects/cloudstack-installation/en/4.6/>> (2016), last access Feb, 2016
- [8] Coyne, L., Gopalakrishnan, S., Sing, J., Redbooks, I.: *IBM Private, Public, and Hybrid Cloud Storage Solutions*. IBM Redbooks (2014), <https://books.google.com.br/books?id=Mjr5AwwAAQBAJ>
- [9] Denton, J.: *Learning OpenStack Networking (Neutron)*. Community experience distilled, Packt Publishing (2014), <https://books.google.com.br/books?id=iXrKBAAAQBAJ>
- [10] Dongarra, J., Luszczek, P., Petitet, A.: *The Linpack Benchmark: Past, Present and future*. eeeee (2001)
- [11] Dukaric, R., Juric, M.B.: *Towards a unified taxonomy and architecture of cloud frameworks*. *Future Generation Computer Systems* 29(5), 1196–1210 (2013)
- [12] Eucalyptus: *Eucalyptus: Open Source Private Cloud Software*, <<https://www.eucalyptus.com/eucalyptus-cloud/iaas>> (2015), last access Mar, 2015
- [13] Fifield, T., Fleming, D., Gentle, A., Hochstein, L., Proulx, J., Toews, E., Topjian, J.: *OpenStack Operations Guide*. O’Reilly Media (2014), <https://books.google.com.br/books?id=jQ5pAwAAQBAJ>
- [14] Hentges, E.L., Thomé, B.R.: *Análise e Comparação de Ferramentas Open Source de Computação em Nuvem para o Modelo de Serviço IaaS*. Master’s thesis, Undergraduate Thesis, Sociedade Educacional Três de Maio (SETREM), Três de Maio, RS, Brazil (December 2013)
- [15] Huang, B., Bauer, M., Katchabaw, M.: *Hpcbench - a Linux-based network benchmark for high performance networks*. In: *High Performance Computing Systems and Applications, 2005. HPCS 2005. 19th International Symposium on*. pp. 65–71 (May 2005)

- [16] Iozone: Iozone File System Benchmark (Official Page) <<http://iozone.org/>> (2006), last access in April, 2015
- [17] Iperf: What is Iperf? (2015), <http://iperf.fr/>, last access in Sept, 2015
- [18] ISO JTC1/SC38 Technical Report: ISO/IEC 17788:2014 Information Technology - Cloud Computing - Overview and Vocabulary (2014), http://www.iso.org/iso/catalogue_detail?csnumber=60544, last access in Sept, 2015
- [19] Kumar, R., Gupta, N., Charu, S., Jain, K., Jangir, S.K.: Open Source Solution for Cloud Computing Platform Using OpenStack. *International Journal of Computer Science and Mobile Computing* 3(5), 89–98 (2014)
- [20] Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., Leaf, D.: NIST cloud computing reference architecture. NIST special publication 500, 292 (2011)
- [21] Mahmood, Z.: Continued Rise of the Cloud: Advances and Trends in Cloud Computing. *Computer Communications and Networks*, Springer London (2014), <https://books.google.com.br/books?id=114gBAAAQBAJ>
- [22] Marinescu, D.: Cloud Computing: Theory and Practice. Elsevier Science (2013), <https://books.google.com.br/books?id=mpcBw10nyIgC>
- [23] Maron, C.A.F., Griebler, D.: Avaliando o Desempenho das Ferramentas de Nuvem Privada OpenStack e OpenNebula. Tech. rep., Laboratory of Advanced Researches on Cloud Computing (LARCC) (2015)
- [24] Mccalpin, J.: The Stream Benchmark) <<https://www.cs.virginia.edu/stream/>> (1996), last access in April, 2015
- [25] Nimbus: About Nimbus, <<http://www.nimbusproject.org/about>> (2015), last access Mar, 2015
- [26] OpenNebula: OpenNebula roadmap <<http://opennebula.org/>> (2015), last access Dec, 2015
- [27] OpenQRM: OpenQRM Community Edition, <<http://www.openqrm-enterprise.com/products/community-edition.html>> (2015), last access Mar, 2015
- [28] OpenStack: OpenStack roadmap <<http://openstack.org/software/roadmap/>> (2015), last access Dec, 2015
- [29] Project, O.: Quickstart: OpenNebula 4.12 on Ubuntu 14.04 and KVM *http://docs.opennebula.org/4.12/design_and_installation/quick_starts/qs_ubuntu_vm* (2015), last access Dec, 2015
- [30] Roveda, D., Vogel, A., Griebler, D.: Understanding, Discussing and Analyzing the OpenNebula's and OpenStack's IaaS Management Layers. *Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação (REABTIC)* 3(1), 15 (August 2015)
- [31] Roveda, D., Vogel, A., Maron, C.A.F., Griebler, D., Schepke, C.: Analisando a Camada de Gerenciamento das Ferramentas CloudStack e OpenStack para Nuvens Privadas. In: 13th Escola Regional de Redes de Computadores (ERRC). Sociedade Brasileira de Computação, Passo Fundo, Brazil (September 2015)

-
- [32] Sabharwal, N.: Apache CloudStack Cloud Computing. Community experience distilled, Packt Publishing (2013), https://books.google.com.br/books?id=46m_ciDiCe8C
- [33] Sosinsky, B.: Cloud Computing Bible. Wiley (2010), <https://books.google.com.br/books?id=hvv2pDEAbOEC>
- [34] Srivastava, S., Ciorba, F.M., Banicescu, I.: Employing a study of the robustness metrics to assess the reliability of dynamic loop scheduling (2011)
- [35] Thome, B., Hentges, E., Griebler, D.: Computação em Nuvem: Análise Comparativa de Ferramentas Open Source para IaaS. In: 11th Escola Regional de Redes de Computadores (ERRC). p. 4. Sociedade Brasileira de Computação, Porto Alegre, RS, Brazil (November 2013)
- [36] Toraldo, G.: OpenNebula 3 Cloud Computing. Community experience distilled, Packt Publishing (2012), <https://books.google.com.br/books?id=W0o0o8dIp9UC>
- [37] Vaquero, L.: Open Source Cloud Computing Systems: Practices and Paradigms: Practices and Paradigms. Premier reference source, Information Science Reference (2012), <https://books.google.com.br/books?id=my5V2JUW0UOC>
- [38] Vogel, A.: Surveying the Robustness and Analyzing the Performance Impact of Open Source Infrastructure as a Service Management Tools. Master's thesis, Undergraduate Thesis, Sociedade Educacional Três de Maio (SETREM), Três de Maio, RS, Brazil (August 2015)
- [39] Vogel, A., Griebler, D., Maron, C.A.F., Schepke, C., Fernandes, L.G.: Private IaaS Clouds: A Comparative Analysis of OpenNebula, CloudStack and OpenStack. In: 24th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). pp. 672–679. IEEE, Heraklion Crete, Greece (February 2016)

A. Instalação das Ferramentas de IaaS

Este Capítulo traz a documentação para instalação de pelo uma instalação de cada ferramenta de IaaS implantada durante a realização do projeto HiPerfCloud no ano de 2015.

Contents

A.1	Instalação da ferramenta OpenStack versão Kilo	2
A.2	Instalação da ferramenta OpenNebula versão 4.12	37
A.3	Instalação da ferramenta CloudStack versão 4.5	42

A.1 Instalação da ferramenta OpenStack versão Kilo



Manual de instalação da ferramenta OpenStack versão Kilo

Autor: Adriano Vogel

Edição:02/2016

Revisão: 1

Apresentação

Este documento apresenta um manual que é resultado de implantações estáveis da ferramenta gerenciadora de infraestrutura de nuvem OpenStack. Nas instalações foi usado o sistema operacional Ubuntu Server 14.04 e os componentes utilizados foram escolhidos no planejamento da infraestrutura, dentre dezenas de componentes disponíveis, esses foram escolhidos para oferecerem os serviços de nuvem:

Nova - gerenciamento das máquinas virtuais

Cinder - gerenciamento dos volumes virtuais LVM para oferecer armazenamento para as máquinas virtuais

Neutron - gerencia as redes virtuais

Horizon - oferece uma interface gráfica via Web para gerenciamento da infraestrutura

Keystone - gerencia a autenticação de serviços, componentes e usuários da nuvem.

Glance - Oferece templates de imagens e discos para criação de máquinas virtuais

Os componentes podem ser instalados e configurados em servidores diferentes com distintas configurações. Esse documento cobrirá uma configuração usando 3 máquinas para a nuvem, pode ser usado um número maior ou menor de máquinas, dependendo sempre da arquitetura escolhida e demandas.

Servidor	Rede de gerenciamento	Rede tunelamento	Rede Externa
Controller	192.168.12.21/24		
Network	192.168.12.22/24	192.168.11.22 / 24	192.168.0.10 / 24
Compute	192.168.12.23/24	192.168.11.23 / 24	

Controller - é o nodo que gerencia toda a infraestrutura da nuvem. Nesse são executados o banco de dados, keystone, glance, o mensageiro RabbitMQ, nova-manager.

Network - Esse nodo foi usado para oferecer redes virtuais e conectividade para as instâncias. Foi usado também como servidor de armazenamento com o componente Cinder.

Compute - É o nodo que vai oferecer os recursos para a execução de máquinas virtuais.

Exemplo do arquivo /etc/hosts

```
# controller
192.168.12.21 controller
# network
192.168.12.22 network
# compute
192.168.12.23 compute
```

A instalação é iniciada no nodo controlador com o servidor NTP para sincronizar o horário com todos os demais nodos.

```
# apt-get install ntp  
# service ntp restart
```

O cliente NTP é recomendado que seja instalado nos demais nodos da nuvem, portanto de acordo com esse tutorial no Network e Compute.

```
# apt-get install ntp
```

O arquivo de configuração NTP precisa ser ajustado para o nodo controller ser o servidor de horário, as demais entradas com o parâmetro "server" devem ser removidas e adicionada a entrada para conectar no nodo da rede local.

```
# nano /etc/ntp.conf
```

Adicionando a entrada do servidor controller no arquivo e salve-o.

```
server controller
```

É necessário ainda reiniciar o serviço NTP para que as mudanças tenham efeito:

```
# service ntp restart
```

Instalação dos pacotes OpenStack

A chave e os repositórios do OpenStack precisam ser adicionados para depois baixar os pacotes dessa versão, através dos comandos:

```
# apt-get install ubuntu-cloud-keyring  
  
# echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu" "trusty-updates/kilo main" >  
/etc/apt/sources.list.d/cloudarchive-kilo.list
```

Ainda, é recomendado uma atualização do sistema:

```
# apt-get update && apt-get dist-upgrade
```

Os passos dessa seção precisam ser repetidos em todos os nodos da nuvem.

Instalando o Banco de Dados MySQL

No funcionamento da nuvem OpenStack o banco de dados é usado para armazenar informações da nuvem. O MySQL precisa ser instalado no nodo controller.

```
# apt-get install mariadb-server python-mysqldb
```

Durante a instalação será requisitada a senha do root, escolha uma senha recomendada e repita.

A instalação cria os arquivos de configuração do banco de dados, porém, para a nuvem funcionar adaptações precisam ser feitas.

Abra o arquivo de configuração:

```
nano /etc/mysql/conf.d/mysqld_openstack.cnf
```

E adicione os seguintes parâmetros no na seção mysqld:

```
bind-address = 192.168.12.21
default-storage-engine = innodb
innodb file per table
collation-server = utf8_general_ci
init-connect = 'SET NAMES utf8'
character-set-server = utf8
```

E reinicie o serviço.

```
# service mysql restart
```

Instalando o mensageiro RabbitMQ

O RabbitMQ é usado pelo OpenStack para coordenar operações e status entre os serviços e componentes, e deve ser instalado no nodo controller.

```
# apt-get install rabbitmq-server
```

Precisa ser adicionado também o usuário do openstack:

```
# rabbitmqctl add user openstack password
```

O último parâmetro “password” precisa ser substituído por uma senha recomendada para o serviço. Permissões específicas para o usuários ainda precisam ser definidas:

```
# rabbitmqctl set permissions openstack ".*" ".*" ".*"
```

Instalando e configurando o Keystone

Esse manual mostra como configurar o keystone no openstack e deve ser feito no nodo controller. Antes de instalar o componente, entradas no banco de dados, usuário keystone e token de permissões precisam ser feitos.

Logue no banco de dados:

```
# mysql -u root -p
```

Crie o banco de dados keystone:

```
CREATE DATABASE keystone;
```

Defina o acesso para o keystone no banco de dados, troque o parâmetro PASSWD por uma outra senha escolhida:

```
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'PASSWD';  
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'PASSWD';
```

Saia do banco de dados com o comando “quit” e gere um token que será usado para proteger o sistema:

```
# openssl rand -hex 10
```

Salve o resultado do comando acima e desabilite a iniciação automática do keystone com o comando:

```
# echo "manual" > /etc/init/keystone.override
```

Após, inicie a instalação do componente keystone.

```
# apt-get install keystone python-openstackclient apache2 libapache2-mod-wsgi memcached  
python-memcache
```

Depois de instalado, adaptações precisam ser aplicadas para o correto funcionamento, abra o arquivo:

```
nano /etc/keystone/keystone.conf
```

E adicione os seguintes parâmetros:

```
[DEFAULT]
```

```
...
```

```
admin_token = 43405b090eda983ddde2 ## troque pelo código que você gerou anteriormente
```

```
verbose = True
```

```
[database]
```

```
...
```

```
connection = mysql://keystone:PASSWD@controller/keystone ## Troque PASSWD pela senha que  
você colocou no keystone
```

```
[memcache]
```

```
...
```

```
servers = localhost:11211
```

```
[token]
```

```
...
```

```
provider = keystone.token.providers.uuid.Provider
```

```
driver = keystone.token.persistence.backends.memcache.Token
```

```
[revoke]
```

```
...
```

```
driver = keystone.contrib.revoke.backends.sql.Revoke
```

Para aplicar as mudanças, execute esse comando:

```
# keystone-manage db sync
```

Configurando o Apache HTTP

Edite o arquivo `/etc/apache2/apache2.conf` e configure o nome correto do servidor.

```
ServerName controller
```

Crie um arquivo

```
# nano /etc/apache2/sites-enabled/wsgi-keystone.conf
```

Cole e salve o seguinte conteúdo:

```
Listen 5000
```

```
Listen 35357
```

```
<VirtualHost *:5000>
```

```
WSGIDaemonProcess keystone-public processes=5 threads=1 user=keystone
display-name=%{GROUP}
WSGIProcessGroup keystone-public
WSGIScriptAlias / /var/www/cgi-bin/keystone/main
WSGIApplicationGroup %{GLOBAL}
WSGIPassAuthorization On
<IfVersion >= 2.4>
ErrorLogFormat "%{cu}t %M"
</IfVersion>
LogLevel info
ErrorLog /var/log/apache2/keystone-error.log
CustomLog /var/log/apache2/keystone-access.log combined
</VirtualHost>
```

```
<VirtualHost *:35357>
WSGIDaemonProcess keystone-admin processes=5 threads=1 user=keystone
display-name=%{GROUP}
WSGIProcessGroup keystone-admin
WSGIScriptAlias / /var/www/cgi-bin/keystone/admin
WSGIApplicationGroup %{GLOBAL}
WSGIPassAuthorization On
<IfVersion >= 2.4>
ErrorLogFormat "%{cu}t %M"
</IfVersion>
LogLevel info
ErrorLog /var/log/apache2/keystone-error.log
CustomLog /var/log/apache2/keystone-access.log combined
</VirtualHost>
```

Crie um diretório necessário:

```
# mkdir -p /var/www/cgi-bin/keystone
```

E baixe os componentes:

```
# curl http://git.openstack.org/cgiit/openstack/keystone/plain/httpd/keystone.py?h=stable/kilo | tee
/var/www/cgi-bin/keystone/main /var/www/cgi-bin/keystone/admin
```

Aplique permissões nos arquivos baixados:

```
# chown -R keystone:keystone /var/www/cgi-bin/keystone
# chmod 755 /var/www/cgi-bin/keystone/*
```

E reinicie o servidor Apache:

```
# service apache2 restart
```

Exporte o código hexadecimal gerado anteriormente para fazer a autenticação:

```
# export OS_TOKEN=43405b090eda983ddde2 ## troque esse token com o seu que foi adicionado  
no arquivo keystone.conf
```

Exporte o login:

```
# export OS_URL=http://controller:35357/v2.0
```

Crie a entidade do serviço de identidade:

```
# openstack service create --name keystone --description "OpenStack Identity" identity
```

```
+-----+-----+  
| Field      | Value                                     ||  
+-----+-----+  
| description | OpenStack Identity                       ||  
| enabled     | True                                      ||  
| id          | c65841b4f8df478cbc19524c09fd9724       ||  
| name        | keystone                                 ||  
| type        | identity                                  ||  
+-----+-----+
```

Verifique a criação:

```
openstack service list  
+-----+-----+-----+  
| ID          | Name      | Type  ||  
+-----+-----+-----+  
| c65841b4f8df478cbc19524c09fd9724 | keystone | identity ||  
+-----+-----+-----+
```

Agora crie a identidade para a API do componente:

```
# openstack endpoint create \  
--publicurl http://controller:5000/v2.0 \  
--internalurl http://controller:5000/v2.0 \  
--adminurl http://controller:35357/v2.0 \  
--region RegionOne \  
identity
```

```

+-----+-----+
| Field      | Value                                     |
+-----+-----+
| adminurl   | http://controller:35357/v2.0           |
| id        | f402a9389d474c13a97a78a30f13c6e5     |
| internalurl | http://controller:5000/v2.0           |
| publicurl  | http://controller:5000/v2.0           |
| region     | RegionOne                              |
| service id | c65841b4f8df478cbc19524c09fd9724     |
| service name | keystone                               |
| service type | identity                               |
+-----+-----+

```

Verifique ainda os detalhes criados.

```

# openstack endpoint list
+-----+-----+-----+-----+
| ID          | Region    | Service Name | Service Type |
+-----+-----+-----+-----+
| f402a9389d474c13a97a78a30f13c6e5 | RegionOne | keystone     | identity     |
+-----+-----+-----+-----+

```

Criando Usuários, projetos e papéis de cada um no OpenStack

```

# openstack project create --description "Admin Project" admin
+-----+-----+
| Field      | Value                                     |
+-----+-----+
| description | Admin Project                           |
| enabled     | True                                     |
| id         | 9b05e6bffdb94c8081d665561d05e31e     |
| name       | admin                                    |
+-----+-----+

```

Crie o usuário admin e coloque uma senha:

```

# openstack user create --password-prompt admin
User Password:
Repeat User Password:
+-----+-----+
| Field      | Value                                     |
+-----+-----+
| email     | None                                     |
| enabled   | True                                     |
| id        | 127a9a6b822a4e3eba69fa54128873cd     |

```



```
| name | admin |
| username | admin |
+-----+-----+
```

Agora crie o papel que usuário admin fará na nuvem:

```
# openstack role create admin
+-----+-----+
| Field | Value |
+-----+-----+
| id | 33af4f957aa34cc79451c23bf014af6f |
| name | admin |
+-----+-----+
```

E usuários do projeto:

```
# openstack role add --project admin --user admin admin
+-----+-----+
| Field | Value |
+-----+-----+
| id | 33af4f957aa34cc79451c23bf014af6f |
| name | admin |
+-----+-----+
```

Crie agora o projeto dos serviços que gerenciam a nuvem:

```
# openstack project create --description "Service Project" service
+-----+-----+
| Field | Value |
+-----+-----+
| description | Service Project |
| enabled | True |
| id | 39e1b9944e564ceb9e71c98623b676cd |
| name | service |
+-----+-----+
```

Crie o projeto demo para os usuários regulares:

```
# openstack user create --password-prompt demo
User Password:
Repeat User Password:
+-----+-----+
| Field | Value |
+-----+-----+
| email | None |
| enabled | True |
| id | 453ce23fa9f347b5baa53210aff7f207 |
| name | demo |
+-----+-----+
```

```
| username | demo |
+-----+-----+
||
```

Agora criei os papéis do usuários:

```
# openstack role create user
+-----+-----+
| Field | Value |
+-----+-----+
| id    | fa78c101a7ed40b19de219e7d3eeda62 |
| name | user |
+-----+-----+
||
```

E o papel dos usuários para o projeto demo:

```
# openstack role add --project demo --user demo user
+-----+-----+
| Field | Value |
+-----+-----+
| id    | fa78c101a7ed40b19de219e7d3eeda62 |
| name | user |
+-----+-----+
||
```

Agora verique o funcionamento das operações com os comandos:

```
# openstack project list
```

```
# openstack user list
```

```
# openstack role list
```

Configurando o Glance no OpenStack

Nessa instalação o serviço de imagem foi implantado juntamente com os demais componentes no servidor que controla a nuvem (controller). Um bom início é criar um script que exporta as variáveis do ambiente e carrega as credenciais.

```
# nano admin-openrc.sh
```

E deve colado o seguinte conteúdo:

```
export OS_PROJECT_DOMAIN_ID=default
export OS_USER_DOMAIN_ID=default
export OS_PROJECT_NAME=admin
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
```

```
export OS_PASSWORD=password #Add a senha do keystone
```

```
export OS_AUTH_URL=http://controller:35357/v3
```

O serviço do glance também usa o banco de dados, o login pode ser feito com o comando:

```
# mysql -u root -p
```

Após o acesso ao banco de dados pode ser criado:

```
CREATE DATABASE glance;
```

E o acesso precisa ser configurado

```
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED BY 'password';
```

```
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' IDENTIFIED BY 'password';
```

Novamente o parâmetro *password* deve ser substituído por uma senha escolhida para o serviço.

Agora carregue as credenciais através do script criado anteriormente:

```
# source admin-openrc.sh
```

E crie o usuário Glance e também crie uma senha que precisa ser repetida:

```
# openstack user create --password-prompt glance
```

Adicione o papel do glance ao usuário admin.

```
# openstack role add --project service --user glance admin
```

Crie a entidade do serviço do glance:

```
# openstack service create --name glance --description "OpenStack Image service" image
```

Ainda é necessário configurar a API e o serviço do glance:

```
# openstack endpoint create \
--publicurl http://controller:9292 \
--internalurl http://controller:9292 \
--adminurl http://controller:9292 \
--region RegionOne \
image
```

Instale os pacotes do glance:

```
# apt-get install glance python-glanceclient
```

É necessário ainda configurar os parâmetros do serviço glance no arquivo `/etc/glance/glance-api.conf`.

```
[DEFAULT]
...
notification_driver = noop
verbose = True

[database]
...
connection = mysql://glance:password@controller/glance
#substitua pela senha do banco de dados glance
[keystone authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = glance
password = password #substitua pela senha do serviço glance
[paste deploy]
...
flavor = keystone

[glance_store]
...
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
```

Configure também o arquivo dos registros do serviço localizado em `/etc/glance/glance-registry.conf`

```
[DEFAULT]
...
notification_driver = noop
verbose = True

[database]
...
connection = mysql://glance:password@controller/glance
#senha do banco de dados glance
```

```
[keystone authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = glance
password = password
# troque pela senha do serviço glance
[paste deploy]
...
flavor = keystone
```

Aplique as mudanças no banco de dados

```
# su -s /bin/sh -c "glance-manage db sync" glance
```

Para concluir reinicie os serviços

```
# service glance-registry restart
# service glance-api restart
```

Configurando o Nova no OpenStack

O serviço *compute* é vital para uma nuvem IaaS pois ele interage com o Keystone para autenticação, serviço de imagens para ter templates e volumes e com os usuários através das interfaces para gerenciamento das máquinas virtuais. Os passos abaixo configuram o nova primeiramente no nó controlador.

O nova também precisa usar o banco de dados, faça login:

```
# mysql -u root -p
```

Crie o banco de dados para o Nova:

```
CREATE DATABASE nova;
```

Aplique as permissões, colocando sua senha no parâmetro *password*:

```
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY 'password';
```

```
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY 'password';
```

Carregue as credenciais do script:

```
# source admin-openrc.sh
```

Crie o usuário do Nova no OpenStack

```
# openstack user create --password-prompt nova
```

Adicione o papel do administrador:

```
# openstack role add --project service --user nova admin
```

E o serviço:

```
# openstack service create --name nova --description "OpenStack Compute" compute
```

Configure a API

```
# openstack endpoint create \  
--publicurl http://controller:8774/v2/%(tenant_id)s \  
--internalurl http://controller:8774/v2/%(tenant_id)s \  
--adminurl http://controller:8774/v2/%(tenant_id)s \  
--region RegionOne \  
compute
```

Agora instale os pacotes do Nova no nodo controller:

```
# apt-get install nova-api nova-cert nova-conductor nova-consoleauth nova-novncproxy  
nova-scheduler python-novaclient
```

Agora edite o arquivo do nova, primeiro abra-o:

```
# nano /etc/nova/nova.conf
```

E modifique as seguintes configurações:

```
[DEFAULT]  
...  
rpc_backend = rabbit  
auth_strategy = keystone
```

```
my_ip = 192.168.12.21
## IP do Controller Node
vncserver listen = 192.168.12.21
## IP do Controller Node
vncserver proxyclient address = 192.168.12.21
## IP do Controller Node
[database]
connection = mysql://nova:password@controller/nova
```

Substitua "password" pela senha do banco de dados nova

```
[oslo_messaging_rabbit]
rabbit host = controller
rabbit userid = openstack
rabbit password = password
```

Substitua "password" pela senha do RabbitMQ

```
[keystone_authtoken]
auth uri = http://controller:5000
auth url = http://controller:35357
auth plugin = password
project domain id = default
user domain id = default
project name = service
username = nova
password = password
```

Substitua "password" pela senha dada ao usuário nova

```
[glance]
host = controller
```

```
[oslo_concurrency]
lock_path = /var/lib/nova/tmp
```

Aplique as mudanças no banco de dados:

```
# su -s /bin/sh -c "nova-manage db sync" nova
```

Para finalizar, reinicie os serviços:

```
# service nova-api restart
# service nova-cert restart
# service nova-consoleauth restart
# service nova-scheduler restart
```

```
# service nova-conductor restart
# service nova-novncproxy restart
```

Configurando o Neutron

O Neutron oferece interfaces virtuais para as instâncias executadas na nuvem. Através de redes e roteadores virtuais oferece funções avançadas para a nuvem como isolamento, tunelamento, firewall, redirecionamento de portas, subnets, VLANs, etc. Inicie a instalação no nodo controlador da nuvem.

Logue no banco de dados

```
# mysql -u root -p
```

Crie o banco de dados para o neutron

```
CREATE DATABASE neutron;
```

Defina as permissões, e substitua novamente o parâmetro *password* por uma senha própria.

```
GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' IDENTIFIED BY 'password';
```

```
GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' IDENTIFIED BY 'password';
```

Também carregue as credenciais do ambiente

```
# source admin-openrc.sh
```

Adicione o usuário e senha do neutron

```
# openstack user create --password-prompt neutron
```

Adicione o papel do admin

```
# openstack role add --project service --user neutron admin
```

Configure o serviço do Neutron

```
# openstack service create --name neutron --description "OpenStack Networking" network
```

E agora configure a API

```
# openstack endpoint create \
--publicurl http://controller:9696 \
--adminurl http://controller:9696 \
--internalurl http://controller:9696 \
```



```
--region RegionOne \
network
```

Agora os pacotes podem ser instalados:

```
# apt-get install neutron-server neutron-plugin-ml2 python-neutronclient
```

E abra o arquivo do neutron localizado em `/etc/neutron/neutron.conf` e modifique as seguintes configurações.

```
[DEFAULT]
verbose = True
rpc_backend = rabbit
auth_strategy = keystone
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
nova_url = http://controller:8774/v2
```

```
[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = password
```

Substitua "password" pela senha do RabbitMQ

```
[database]
connection = mysql://neutron:password@controller/neutron
```

Substitua "password" pela senha do banco de dados neutron

```
[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = neutron
password = password
```

Substitua "password" pela senha dada ao usuário neutron

```
[nova]
auth_url = http://controller:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
region_name = RegionOne
project_name = service
```

```
username = nova
password = password
```

Substitua "password" pela senha dada ao usuário nova

Ainda é necessário configurar o plugin ML2, no arquivo `/etc/neutron/plugins/ml2/ml2_conf.ini`
Modificando o seguinte:

```
[ml2]
type_drivers = flat,vlan,gre,vxlan
tenant_network_types = gre
mechanism_drivers = openvswitch
```

```
[ml2 type gre]
tunnel_id_ranges = 1:1000
```

```
[securitygroup]
enable_security_group = True
enable_ipset = True
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
```

Configure o serviço do Nova (`/etc/nova/nova.conf`) para usar o serviço de rede Neutron, adicionando as seguintes configurações:

```
[DEFAULT]
network_api_class = nova.network.neutronv2.api.API
security_group_api = neutron
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSIfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver
[neutron]
url = http://controller:9696
auth_strategy = keystone
admin_auth_url = http://controller:35357/v2.0
admin_tenant_name = service
admin_username = neutron
admin_password = password
```

Substitua "password" pela senha dada ao usuário neutron

Aplique as mudanças no banco de dados:

```
# su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
```

E reinicie os serviços do Neutron

```
# service nova-api restart
# service neutron-server restart
```

Verifique o status com o comando:

```
# neutron ext-list
```

Configurando o Neutron no nodo de rede (Network)

Na seção anterior o Neutron foi configurado no controller, agora é necessário fazer a instalação no servidor da rede. Primeiro, adicione os repositórios e atualize o sistema.

```
# apt-get install ubuntu-cloud-keyring
```

```
# echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu \"trusty-updates/kilo main\" >  
/etc/apt/sources.list.d/cloudarchive-kilo.list
```

```
# apt-get update
```

Esse servidor vai oferecer conectividade para as VMs, o compartilhamento da conexão precisa ser configurado no arquivo `/etc/sysctl.conf` adicionando os seguintes parâmetros:

```
net.ipv4.ip_forward=1  
net.ipv4.conf.all.rp_filter=0  
net.ipv4.conf.default.rp_filter=0
```

E aplique as mudanças:

```
# sysctl -p
```

Instale os componentes no nodo de rede:

```
# apt-get install neutron-plugin-ml2 neutron-plugin-openvswitch-agent neutron-l3-agent  
neutron-dhcp-agent neutron-metadata-agent
```

Configure o arquivo principal do Neutron (`/etc/neutron/neutron.conf`) adicionando os seguintes parâmetros:

```
[DEFAULT]  
rpc_backend = rabbit  
core_plugin = ml2  
service_plugins = router  
allow_overlapping_ips = True  
auth_strategy = keystone  
verbose = True
```

```
[oslo_messaging_rabbit]  
rabbit_host = controller  
rabbit_userid = openstack
```

```
rabbit_password = password
```

Substitua "password" pela senha do RabbitMQ

```
[database]
#connection = sqlite:///var/lib/neutron/neutron.sqlite
## Comente a linha acima
[keystone_auth_token]
auth_uri = http://controller:5000
auth_url = http://controller:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = neutron
password = password
```

Substitua "password" pela senha do usuário neutron

Configure também o plugin ML2 no arquivo /etc/neutron/plugins/ml2/ml2_conf.ini e adicione as seguintes configs:

```
[ml2]
type_drivers = flat,vlan,gre,vxlan
tenant_network_types = gre
mechanism_drivers = openvswitch
```

```
[ml2_type_flat]
flat_networks = external
```

```
[ml2_type_gre]
tunnel_id_ranges = 1:1000
```

```
[securitygroup]
enable_security_group = True
enable_ipset = True
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
```

```
[ovs]
local_ip = 192.168.11.22
## Ip da interface do tunel
bridge_mappings = external:br-ex
```

```
[agent]
tunnel_types = gre
```

Nesse servidor também é preciso configurar o roteamento (nível 3) para as redes virtuais no arquivo `/etc/neutron/l3_agent.ini` modificando a seção:

```
[DEFAULT]
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
external_network_bridge =
router_delete_namespaces = True
verbose = True
```

Também o servidor DHCP precisa ser habilitado através da configuração no arquivo `/etc/neutron/dhcp_agent.ini`

```
[DEFAULT]
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
dhcp_delete_namespaces = True
verbose = True
```

Configure ainda o agente de metadados no arquivo `/etc/neutron/metadata_agent.ini`

```
[DEFAULT]
verbose = True
auth_uri = http://controller:5000
auth_url = http://controller:35357
auth_region = RegionOne
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = neutron
password = password
```

Substitua "password" pela senha dada ao usuário neutron

```
nova_metadata_ip = controller
metadata_proxy_shared_secret = 26f008fb8c504b393df3
```

Substitua "26f008fb8c504b393df3" por uma chave para o proxy

Agora volte ao nodo principal da nuvem (controller) e adicione ao arquivo `/etc/nova/nova.conf` na seção os seguintes parâmetros:

```
[neutron]
```

```
service metadata proxy = True
```

```
metadata proxy shared secret = 26f008fb8c504b393df3
```

Substitua "26f008fb8c504b393df3" pela chave escolhida e adicionada acima.

Reinicie a API do nova (service nova-api restart)

Reinicie o plugin Open VSwitch no nodo de rede:

```
# service openvswitch-switch restart
```

Adicione a interface externa bridge ao nodo de rede

```
# ovs-vsctl add-br br-ex
```

Adicione ainda porta para a bridge externa, nesse caso a eth2 (coloque o nome da interface do seu ambiente):

```
# ovs-vsctl add-port br-ex eth2
```

Reinicie os serviços de rede

```
# service neutron-plugin-openvswitch-agent restart
```

```
# service neutron-l3-agent restart
```

```
# service neutron-dhcp-agent restart
```

```
# service neutron-metadata-agent restart
```

Verifique as operações no nodo controller

```
# source admin-openrc.sh
```

O resultado de listar os agentes deve ser semelhante a isso

```
# neutron agent-list
```

```
+-----+-----+-----+-----+-----+-----+
| id                | agent_type | host | alive | admin_state_up | binary |
+-----+-----+-----+-----+-----+-----+
| 23da3f95-b81b-4426-9d7a-d5cbfc5241c0 | Metadata agent |      |      | network | :- ) | True |
neutron-metadata-agent |
| 4217b0c0-fbd4-47d9-bc22-5187f09d958a | DHCP agent |      |      | network | :- ) | True |
neutron-dhcp-agent |
| b4cf95cd-2eba-4c69-baa6-ae8832384e40 | Open vSwitch agent | network | :- ) | True |
neutron-openvswitch-agent |
| d9e174be-e719-4f05-ad05-bc444eb97df5 | L3 agent |      |      | network | :- ) | True |
neutron-l3-agent |
+-----+-----+-----+-----+-----+-----+
```

Configurando o OpenStack Cinder no Controller

O Cinder oferece armazenamento em blocos para o ambiente de nuvem. O início da configuração é o no nodo controller, logando no banco de dados e criando as entradas para o cinder.

```
# mysql -u root -p
```

```
CREATE DATABASE cinder;
```

Defina também as permissões, sempre colocado uma senha própria no parâmetro.

```
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' IDENTIFIED BY 'password';
```

```
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' IDENTIFIED BY 'password';
```

Carregue as credenciais do script exportando as variáveis:

```
# source admin-openrc.sh
```

Adicione o usuário do cinder

```
# openstack user create --password-prompt cinder
```

E o admin:

```
# openstack role add --project service --user cinder admin
```

Crie as entidade do serviço

```
# openstack service create --name cinder --description "OpenStack Block Storage" volume
```

```
# openstack service create --name cinderv2 --description "OpenStack Block Storage" volumev2
```

Adicione a API de armazenamento em blocos:

```
# openstack endpoint create --publicurl http://controller:8776/v2/%(tenant_id)s --internalurl  
http://controller:8776/v2/%(tenant_id)s --adminurl http://controller:8776/v2/%(tenant_id)s --region  
RegionOne volume
```

```
# openstack endpoint create --publicurl http://controller:8776/v2/%(tenant_id)s --internalurl  
http://controller:8776/v2/%(tenant_id)s --adminurl http://controller:8776/v2/%(tenant_id)s --region  
RegionOne volumev2
```

Instale o componente:

```
# apt-get install cinder-api cinder-scheduler python-cinderclient
```

Modifique as seguintes configurações no arquivo do cinder (*/etc/cinder/cinder.conf*)

```
[database]  
connection = mysql://cinder:password@controller/cinder
```

Substitua "password" pela senha dada ao usuário do banco de dados do cinder

```
[DEFAULT]  
rpc_backend = rabbit  
auth_strategy = keystone  
verbose = True  
my_ip = 192.168.12.21
```

```
[oslo_messaging_rabbit]  
rabbit_host = controller  
rabbit_userid = openstack  
rabbit_password = password
```

Substitua "password" pela senha dada ao RabbitMQ

```
[keystone_authtoken]  
auth_uri = http://controller:5000  
auth_url = http://controller:35357  
auth_plugin = password  
project_domain_id = default  
user_domain_id = default  
project_name = service  
username = cinder  
password = password
```

Substitua "password" pela senha dada ao usuário cinder

Comente ou remova qualquer outra configuração que tiver em [keystone_authtoken]

```
[oslo_concurrency]  
lock_path = /var/lock/cinder
```


Remova a configuração de lock_path na seção [DEFAULT]

Aplica as mudanças no banco de dados

```
# su -s /bin/sh -c "cinder-manage db sync" cinder
```

Reinicie os serviços:

```
# service cinder-scheduler restart
```

```
# service cinder-api restart
```

Configurando o Cinder no nodo de armazenamento

O servidor do cinder é onde ocorrerá exatamente o armazenamento dos dados da nuvem. Nessa implantação no nodo Network foi usado também como servidor de armazenamento, mas é recomendado que seja usado um servidor específico para o armazenamento. O armazenamento em bloco será configurado para a partição /dev/sdb que já contém a partição /dev/sdb1 criada.

Comece instalando e configurando o LVM

```
# apt-get install lvm2
```

Crie o volume físico na partição escolhida:

```
# pvcreate /dev/sdb1
```

Agora crie o grupo virtual associado ao volume físico:

```
# vgcreate vg_cinder /dev/sdb1
```

Agora edite o arquivo de configuração do lvm (*/etc/lvm/lvm.conf*) para que ele escaneie pela partição correta:

De

```
filter = [ "a./" ]
```

Para

```
filter = [ "a/sdb/", "r./" ]
```

Instale e configure os componentes do Cinder

```
# apt-get install cinder-volume python-mysqldb
```

Edite o arquivo de configuração do Cinder (*/etc/cinder/cinder.conf*) e altere os seguintes parâmetros.

```
[DEFAULT]
```

```
rpc_backend = rabbit
```

```
auth_strategy = keystone
```

```
my_ip = 192.168.12.24
```

```
## IP da rede de gerenciamento no nodo de armazenamento
```

```
enabled_backends = lvm
```

```
glance_host = controller
```

```
verbose = True
```

```
[database]
```

```
connection = mysql://cinder:password@controller/cinder
```

```
## Substitua "password" pela senha dada ao usuário do banco de dados do cinder
```

```
[oslo_messaging_rabbit]
```

```
rabbit_host = controller
```

```
rabbit_userid = openstack
```

```
rabbit_password = password
```

```
## Substitua "password" pela senha do serviço RabbitMQ
```

```
[keystone_authtoken]
```

```
auth_uri = http://controller:5000
```

```
auth_url = http://controller:35357
```

```
auth_plugin = password
```

```
project_domain_id = default
```

```
user_domain_id = default
```

```
project_name = service
```

```
username = cinder
```

```
password = password
```

```
## Substitua "password" pela senha do usuário cinder
```

```
[lvm]
```

```
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
```

```
volume_group = vg_cinder
```

```
iscsi_protocol = iscsi
```

```
iscsi_helper = tgtadm
```

```
[oslo_concurrency]
```

```
lock_path = /var/lock/cinder
```

Remova a configuração de lock_path na seção [DEFAULT]

Reinicie os serviços de armazenamento

```
# service tgt restart
```

```
# service cinder-volume restart
```

Para verificar a operação, se conecte ao nó controller e execute:

```
# echo "export OS_VOLUME_API_VERSION=2" | tee -a admin-openrc.sh demo-openrc.sh
```

Exporte as credenciais

```
# source admin-openrc.sh
```

E verifique se os serviços adicionados estão funcionando corretamente:

```
# cinder service-list
```

Adicionando um nó computacional a nuvem

Os *compute nodes* são os servidores que executam as máquinas virtuais. Nessa instalação usamos o nó Compute para instalar esses serviços, em um ambiente real diversos nós podem ser adicionados seguindo esse manual.

Instale o servidor NTP ao nó:

```
# apt-get install ntp
```

Edite o arquivo de configuração do NTP (*/etc/ntp.conf*), comente todas as linhas com o parâmetro "server" e adicione a seguinte para o nó sincronizar o horário com o servidor controlador da nuvem:

```
server controller
```

E reinicie o serviço NTP:

```
# service ntp restart
```

Instale ainda os pacotes OpenStack e atualize o sistema:

```
# apt-get install ubuntu-cloud-keyring
```

```
# echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu" "trusty-updates/kilo main" >  
/etc/apt/sources.list.d/cloudarchive-kilo.list
```

```
# apt-get update
```

E instale os pacotes necessários para o funcionamento do componente Nova:

```
# apt-get install nova-compute sysfsutils
```

Edite o arquivo de configuração do nova (*/etc/nova/nova.conf*) com os seguintes parâmetros:

```
[DEFAULT]
rpc_backend = rabbit
auth_strategy = keystone
my_ip = 192.168.12.23
## IP da rede de gerenciamento do nodo
vnc_enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = 192.168.12.23
## IP da rede de gerenciamento do nodo
novncproxy_base_url = http://controller:6080/vnc_auto.html
```

```
[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = password
## Substitua "password" pela senha do serviço RabbitMQ
```

```
[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = nova
password = password
```

Substitua "password" pela senha do usuário nova no serviço de identidade do nodo controlador

```
[glance]
host = controller
```

```
[oslo_concurrency]
lock_path = /var/lib/nova/tmp
```

E reinicie o serviço:

```
# service nova-compute restart
```

Verifique se o nodo foi adicionado logando no nodo controller e executando o comando:

```
# nova service-list
```

Para o nodo usar a rede do Neutron, adicione ao arquivo de configuração do nova (*/etc/nova/nova.conf*) os seguintes parâmetros:

```
[DEFAULT]
```

```
...
```

```
network_api_class = nova.network.neutronv2.api.API
```

```
security_group_api = neutron
```

```
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSIfaceDriver
```

```
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

```
[neutron]
```

```
url = http://controller:9696
```

```
auth_strategy = keystone
```

```
admin_auth_url = http://controller:35357/v2.0
```

```
admin_tenant_name = service
```

```
admin_username = neutron
```

```
admin_password = password
```

Substitua "password" pela senha do usuário neutron no serviço de identidade do nodo controlador

E reinicie os serviços no nodo:

```
# service nova-compute restart
```

```
# service neutron-plugin-openvswitch-agent restart
```

Adicionando a Dashboard OpenStack

Esse procedimento precisa ser realizado no nodo controller, execute:

```
# apt-get install openstack-dashboard
```

E edite o arquivo de configuração da interface web localizando em */etc/openstack-dashboard/local_settings.py* :

coloque o nome do nodo

```
OPENSTACK_HOST = "controller"
```

Autorize todos os nodos para acessarem

```
ALLOWED_HOSTS = '*'
```

```
CACHES = {  
    'default': {  
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',  
        'LOCATION': '127.0.0.1:11211',  
    }  
}
```

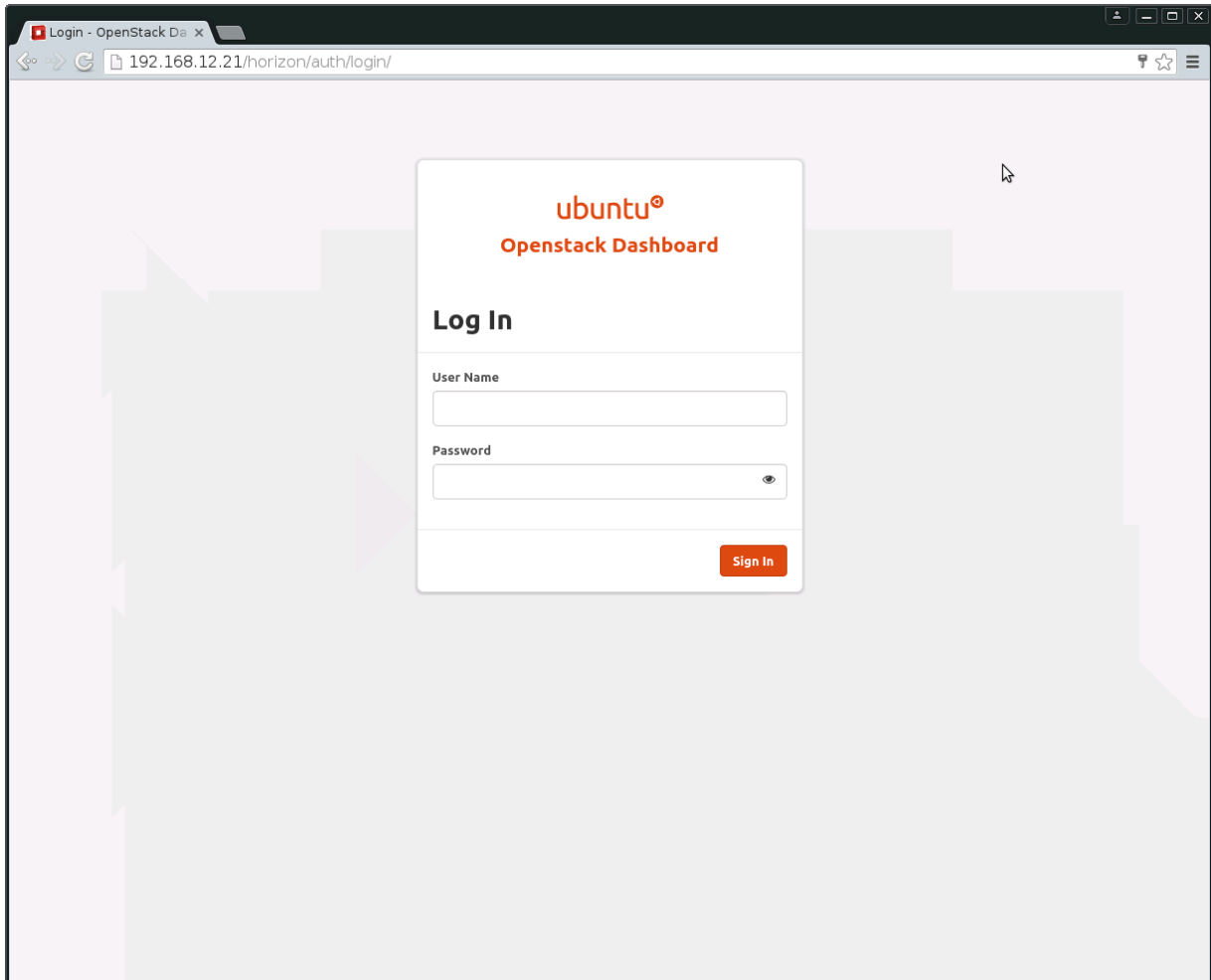
autorização padrão da interface

```
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
```

Reinicie o serviço do apache:

```
# service apache2 restart
```

É possível acessar a interface web no endereço: <http://ip-address/horizon>



Verifique os servidores adicionados para executarem VMs:

The screenshot displays the 'All Hypervisors' page in the OpenStack Horizon interface. The page title is 'All Hypervisors' and the sub-section is 'Hypervisor Summary'. Three pie charts show the usage of resources:

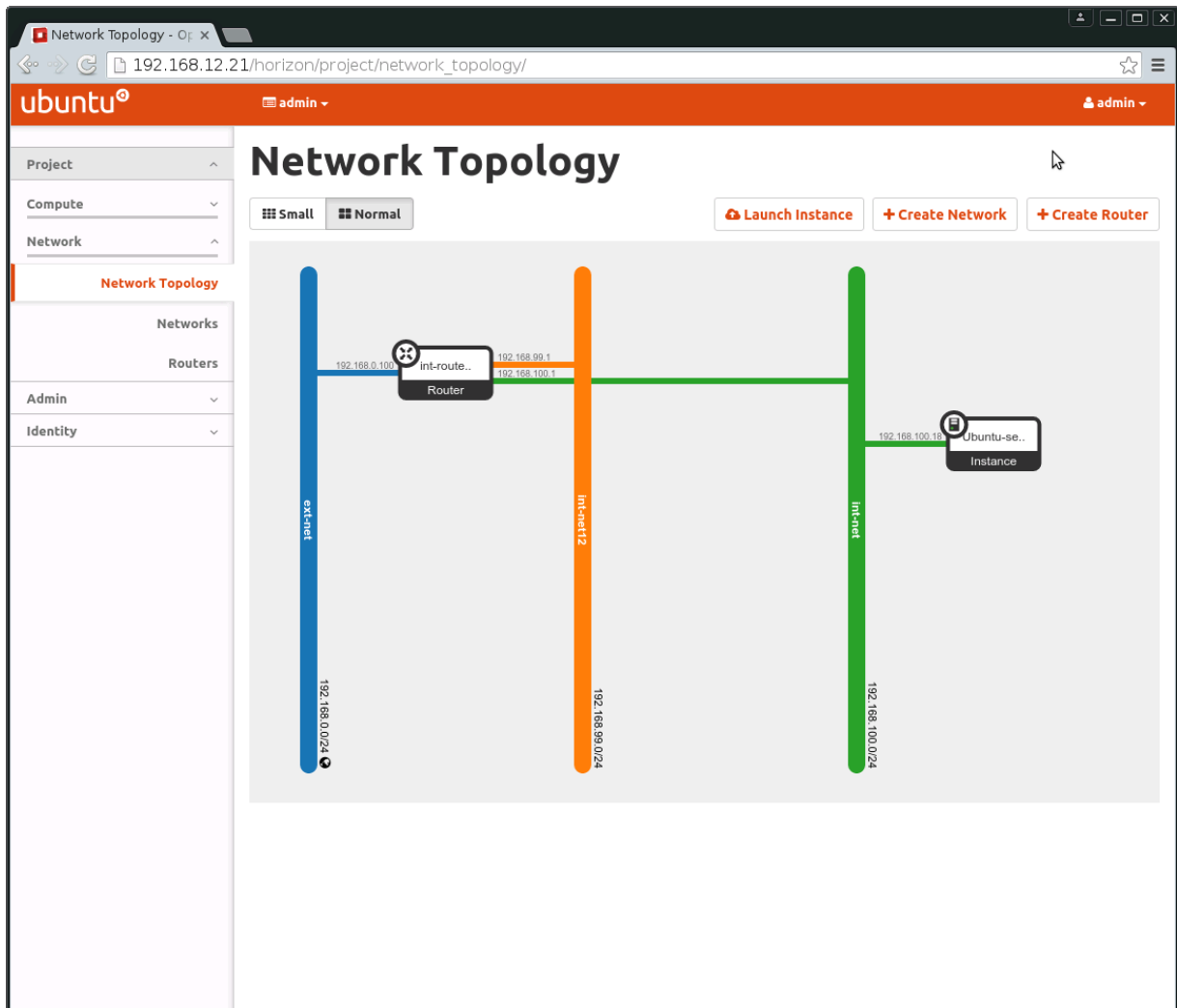
- VCPU Usage: Used 4 of 8
- Memory Usage: Used 6GB of 11.4GB
- Local Disk Usage: Used 126GB of 225GB

Below the charts, there is a table titled 'Compute Host' showing the details of the hypervisors:

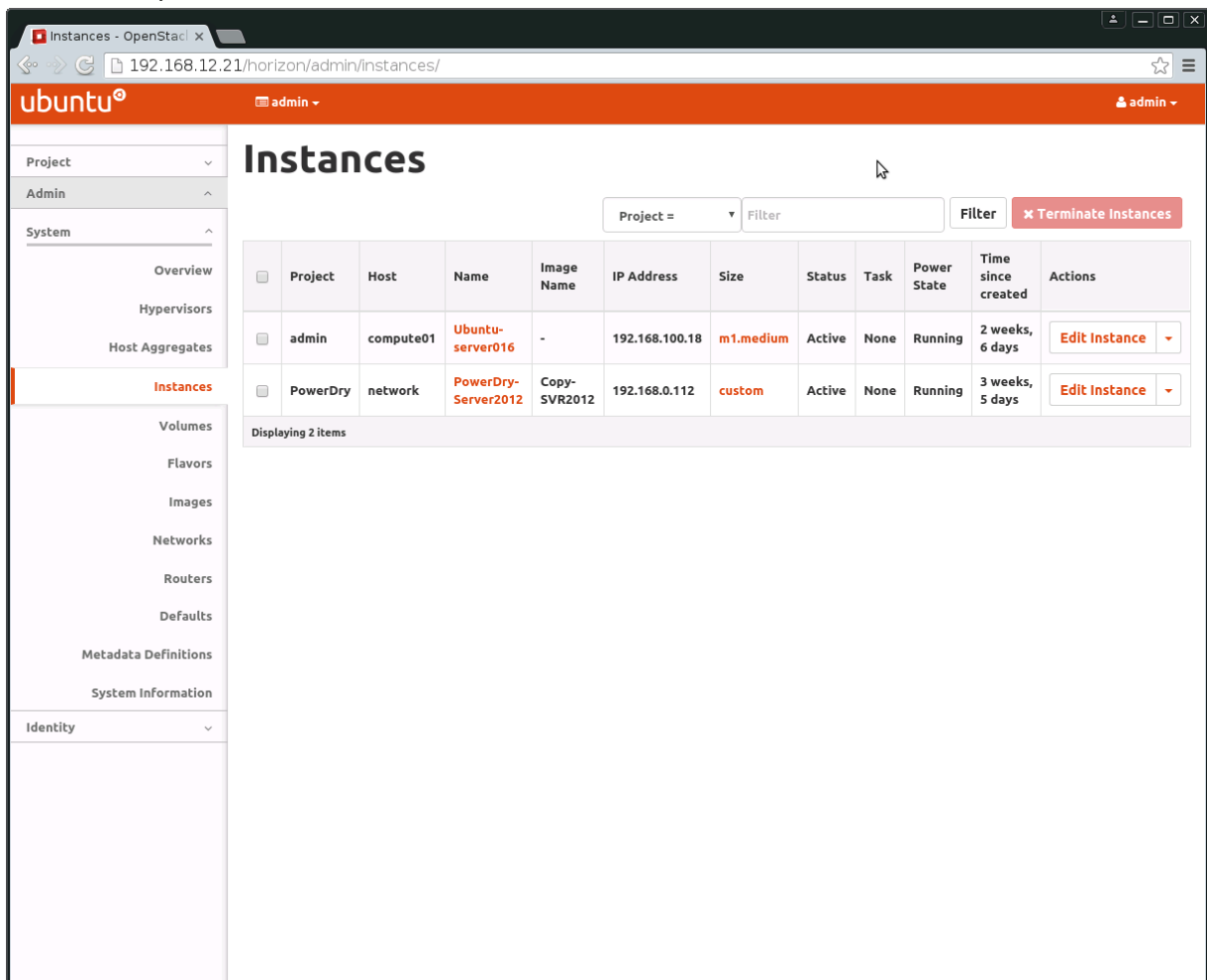
Hostname	Type	VCPUs (used)	VCPUs (total)	RAM (used)	RAM (total)	Local Storage (used)	Local Storage (total)	Instances
controller	QEMU	0	4	512MB	3.6GB	0Bytes	100GB	0
network	QEMU	2	4	1.5GB	3.6GB	86GB	100GB	1
fordcorcel	QEMU	0	4	512MB	7.8GB	0Bytes	125GB	0
compute01	QEMU	2	4	4.5GB	7.8GB	40GB	125GB	1

Displaying 4 items

Exemplo da topologia da rede virtual criada:



Lista da máquinas virtuais criadas:



The screenshot shows the OpenStack Horizon web interface. The browser address bar displays `192.168.12.21/horizon/admin/instances/`. The page title is "Instances". On the left, there is a navigation sidebar with categories: Project, Admin, System, Overview, Hypervisors, Host Aggregates, Instances (highlighted), Volumes, Flavors, Images, Networks, Routers, Defaults, Metadata Definitions, System Information, and Identity. The main content area features a table of instances with the following columns: Project, Host, Name, Image Name, IP Address, Size, Status, Task, Power State, Time since created, and Actions. Two instances are listed: one from project 'admin' on host 'compute01' with name 'Ubuntu-server016', and another from project 'PowerDry' on host 'network' with name 'PowerDry-Server2012'. Both are in 'Active' status and 'Running' power state. A 'Filter' box and a 'Terminate Instances' button are located above the table.

<input type="checkbox"/>	Project	Host	Name	Image Name	IP Address	Size	Status	Task	Power State	Time since created	Actions
<input type="checkbox"/>	admin	compute01	Ubuntu-server016	-	192.168.100.18	m1.medium	Active	None	Running	2 weeks, 6 days	Edit Instance
<input type="checkbox"/>	PowerDry	network	PowerDry-Server2012	Copy-SVR2012	192.168.0.112	custom	Active	None	Running	3 weeks, 5 days	Edit Instance

Displaying 2 items

A.2 Instalação da ferramenta OpenNebula versão 4.12



Manual de instalação da ferramenta OpenNebula versão 4.12

Autor: Adriano Vogel

Edição:02/2016

Revisão: 1

Apresentação

Este tutorial apresenta a configuração da ferramenta gerenciadora de infraestrutura de nuvem OpenNebula. Os pacotes e serviços foram instalados no sistema operacional Ubuntu Server 14.04, em outras distribuições adaptações precisam ser feitas, esse documento não cobre tais aspectos.

A arquitetura dessa implantação de nuvem é formada por um servidor chamado Frontend que controla a nuvem e nodos, que fornecem os recursos computacionais para a execução das máquinas virtuais.

Configurando o Frontend

Um dos primeiros passos é adicionar as chaves dos repositórios OpenNebula e também as entradas OpenNebula no arquivo *sources.list*.

```
# wget -q -O- http://downloads.opennebula.org/repo/Ubuntu/repo.key | apt-key add -  
# echo "deb http://downloads.opennebula.org/repo/4.12/Ubuntu/14.04/ stable opennebula"  
\  
> /etc/apt/sources.list.d/opennebula.list
```

Os dois comandos acima devem ser executados com privilégios de *root*, e em seguida uma atualização do sistema é recomendada através do comando abaixo

```
# apt-get update
```

Após adicionar os repositórios e atualizar o sistema os pacotes do OpenNebula podem ser baixados e compilados:

```
# apt-get install opennebula opennebula-sunstone nfs-kernel-server
```

OpenNebula baixa os *scripts* e *drivers* necessários, o pacote *opennebula-sunstone* é a interface gráfica para gerenciamento da infraestrutura e NFS é usado para oferecer volumes virtuais compartilhados através da rede.

Para a interface ser acessível pelos hosts da rede o arquivo de configuração precisa ser editado através do arquivo */etc/one/sunstone-server.conf*, mudando o parâmetro *:host:* *127.0.0.1* para *:host: 0.0.0.0*. Para aplicar as mudanças o servidor precisa ser reiniciado:

```
# /etc/init.d/opennebula-sunstone restart
```

Para o funcionamento do serviço NFS requer a configuração dos diretórios a serem exportados. Nessa instalação todo o conteúdo do diretório `/var/lib/one/` do servidor Frontend vai ser exportados e montado nos nodos da rede. Para isso, uma entrada precisa ser adicionada no arquivo `/etc/exports`:

```
/var/lib/one/ *(rw, sync, no_subtree_check, root_squash)
```

Para aplicar as mudanças e exportar o diretório o serviço NFS precisa ser reiniciado:

```
# service nfs-kernel-server restart
```

O funcionamento distribuído da ferramenta requer acesso SSH sem senha entre os nodos do cluster da nuvem, o usuário `oneadmin` é criado automaticamente e este é que realiza as tarefas administrativas através dos seus serviços. É possível logar nesse usuário com o comando:

```
# su - oneadmin
```

Após é possível copiar as chaves SSH que serão compartilhadas:

```
$ cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
```

É recomendando ainda a configuração para evitar a checagem dos hosts no momento da conexão, no arquivo conhecido como `known_hosts`. Isso pode ser desabilitado:

```
$ cat << EOT > ~/.ssh/config
Host *
    StrictHostKeyChecking no
    UserKnownHostsFile /dev/null
EOT
```

As chaves SSH precisam ter suas permissões alteradas:

```
$ chmod 600 ~/.ssh/config
```

Instalação nos nodos

Para a execução de máquinas virtuais com desempenho aceitável é recomendado o suporte dos processadores para acelerados de hardware e virtualização, que pode ser verificado com o comando:

```
grep -E 'svm|vmx' /proc/cpuinfo
```

Se o resultado for 1 ou maior é o reflexo do número de núcleos que possuem tal suporte. A instalação OpenNebula em um nodo necessita as chaves dos repositórios OpenNebula e também as entradas OpenNebula no arquivo *sources.list*.

```
# wget -q -O- http://downloads.opennebula.org/repo/Ubuntu/repo.key | apt-key add -
# echo "deb http://downloads.opennebula.org/repo/4.12/Ubuntu/14.04/ stable opennebula"
\
> /etc/apt/sources.list.d/opennebula.list
```

Os dois comandos acima devem ser executados com privilégios de *root*, e em seguida uma atualização do sistema é recomendada através do comando abaixo

```
# apt-get update
```

Após adicionar os repositórios e atualizar o sistema os pacotes do OpenNebula podem ser baixados e compilados:

```
apt-get install opennebula-node nfs-common bridge-utils
```

O pacote *opennebula-node* baixa os componentes necessários para a execução e gerenciamento das máquinas virtuais (Ex: KVM, QEMU, Libvirt). o *nfs-common* é o cliente NFS e o *bridge-utils* permite a criação das interfaces linux em bridge. Para oferecer conectividade para as máquinas virtuais as bridges são usadas. Nesse exemplo, diversas interfaces virtuais são adicionadas sob a mesma interface física eth0. Inicialmente o o arquivo */etc/network/interfaces* precisa ser editado, como esse exemplo:

```
auto lo
iface lo inet loopback

auto br0
iface br0 inet static
    address 11.11.11.22
    network 11.11.11.0
    netmask 255.255.255.0
    broadcast 192.168.0.255
    gateway 11.11.11.1
    bridge_ports eth0
    bridge_fd 9
    bridge_hello 2
```

```
bridge_maxage 12
```

```
bridge_stp off
```

Após a rede precisa ser reiniciada para que as mudanças tomem efeito:

```
/etc/init.d/networking restart
```

Outro ponto importante é montar os diretórios compartilhados no Frontend da nuvem. O arquivo localizado em `/etc/fstab` recebe uma entrada com o IP do servidor, o diretório e parâmetros do diretório.

```
11.11.11.21:/var/lib/one/ /var/lib/one/ nfs soft,intr,rsize=8192,wsiz=8192,noauto
```

Após o diretório pode ser montado com o comando:

```
# mount /var/lib/one/
```

Nos nodos também é criado o usuário `oneadmin` que gerencia os recursos. Para a execução das máquinas virtuais, esse usuário precisa ser configurado nos parâmetros da tecnologia de virtualização:

```
# cat << EOT > /etc/libvirt/qemu.conf
```

```
user = "oneadmin"
```

```
group = "oneadmin"
```

```
dynamic_ownership = 0
```

```
EOT
```

O serviço `libvirt` precisa ser reiniciado para que as mudanças sejam aplicadas:

```
service libvirt-bin restart
```

Após a nuvem pode ser acessada via web no endereço: <http://IP-frontend:9869>. Após o login a infraestrutura pode ser adicionada e é possível criar máquinas virtuais.

A.3 Instalação da ferramenta CloudStack versão 4.5



Manual de instalação da ferramenta CloudStack versão 4.5

Autor: Adriano Vogel

Edição:02/2016

Revisão: 1

Apresentação

CloudStack é uma ferramenta de gerenciamento de infraestrutura de nuvem que se tornou popular pela arquitetura flexível e suporta para alta disponibilidade. Esse documento apresenta a instalação do nodo controlador da nuvem, através da configuração do *cloudstack-management*, usando o mesmo servidor para armazenamento NFS. É mostrada também a configuração para adicionar nodos computacionais, que são instalados com o *cloudstack-agent* e alocam os recursos para a execução das máquinas virtuais.

Instalação do Gerente CloudStack

Configure a resolução de nomes, nome do domínio e o nome do host com o seguinte exemplo de comando:

```
HOSTNAME=dodge1-cloudstack
DOMAINNAME=dodge1.larcc
IP=`ip addr show eth0 | grep 'inet ' | awk '{print $2}' | sed -e 's,/,*,'`
sed -i -e "s/^127.0.1.1.*/$IP $HOSTNAME.$DOMAINNAME $HOSTNAME/" /etc/hosts
cat /etc/hosts
hostname --fqdn
dig `hostname --fqdn` @8.8.8.8
```

Esse comando define o nome como `dodge1-cloudstack`, o domínio para `dodge1.larcc` e também atualiza o arquivo `/etc/hosts`.

Instale também servidor OpenNTP para manter os horários atualizados:

```
apt-get install openntp
```

O próximo passo é configurar a rede no servidor, a ferramenta CloudStack usa as bridges linux para criar interfaces virtuais para as instâncias. Instale os pacotes necessários com o comando:

```
apt-get install bridge-utils
```

Faça ainda uma cópia do arquivo de interfaces, pois ele será modificado:

```
cp /etc/network/interfaces /etc/network/interfaces.orig
```

O comando abaixo fixa o IP `11.11.11.30` ao servidor, configura a máscara, gateway e bridge `cloudbr0` ligada a interface física `eth0`.

```
cat >/etc/network/interfaces <<EOM
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet manual

# Public network
auto cloudbr0
iface cloudbr0 inet static
    address 11.11.11.30
    netmask 255.255.255.0
    gateway 11.11.11.20
    dns-nameservers 8.8.8.8
    bridge_ports eth0
    bridge_fd 5
    bridge_stp off
    bridge_maxwait 1

# Private network
auto cloudbr1
iface cloudbr1 inet manual
    bridge_ports none
    bridge_fd 5
    bridge_stp off
    bridge_maxwait 1
EOM
```

Agora reinicie o sistema operacional para aplicar as mudanças nas interfaces.

Depois é possível iniciar a instalação dos pacotes CloudStack, primeiro adicionado o repositório, as chaves APT e atualizando o sistema:

```
cat >/etc/apt/sources.list.d/cloudstack.list <<EOM
deb http://cloudstack.apt-get.eu/ubuntu trusty 4.4
EOM
wget -O - http://cloudstack.apt-get.eu/release.asc|apt-key add -
apt-get update
```

Instale o gerente CloudStack

```
apt-get --yes install cloudstack-management
```

Após é necessário instalar também o banco de dados MySQL:

```
apt-get --yes install mysql-server
```

Será solicitada uma senha para o administrador do banco de dados. Escolha uma senha forte, para fins de teste nesse documento será colocada a senha *mydbpasswd00*

Configure parâmetros do banco de dados e reinicie-o:

```
cat >>/etc/mysql/conf.d/cloudstack.cnf <<EOM
[mysqld]
innodb_rollback_on_timeout=1
innodb_lock_wait_timeout=600
max_connections=350
log-bin=mysql-bin
binlog-format = 'ROW'
EOM
```

```
service mysql restart
```

O banco de dados precisa ainda ser configurado para o cloudstack funcionar. O comando abaixo usa credenciais de teste, substitua pelas próprias e configura o banco de dados com o script `cloudstack-setup-database`.

```
# mysql root password: mydbpasswd00
# cloud user password: mycupasswd11
# management server key: mymskey44
# database key: mydbkey00
cloudstack-setup-databases cloud:mycupasswd11@localhost \
--deploy-as=root:mydbpasswd00 \
-e file \
-m mymskey44 \
-k mydbkey00
```

A próxima parte é configurar o armazenamento para a nuvem, que nesse tutorial será através do NFS. Crie os diretórios que serão exportados. O primary é onde ficarão os volumes das máquinas virtuais, enquanto no secondary ficam as ISOs, templates e snapshots.

```
mkdir -p /export/primary /export/secondary
```

Instale o servidor NFS:

```
apt-get install nfs-kernel-server
```

Adicione o diretório /export ao arquivo de configuração do NFS (/etc/exports), que é como serão exportados:

```
cat >>/etc/exports <<EOM
/export *(rw,async,no root squash,no subtree check)
EOM
```

E aplique as mudanças, para assim exportar o diretório adicionado:

```
exportfs -a
```

É necessário ainda instalar o cliente NFS, que faz a montagem dos diretórios:

```
apt-get install nfs-common
```

Configure a inicialização e portas usadas pelo serviço NFS:

```
cp /etc/default/nfs-common /etc/default/nfs-common.orig
sed -i 'NEED_STATD=/ a NEED_STATD=yes' /etc/default/nfs-common
sed -i '/STATDOPTS=/ a STATDOPTS="--port 662 --outgoing-port 2020"
/etc/default/nfs-common
diff -du /etc/default/nfs-common.orig /etc/default/nfs-common
```

Configure o lockd:

```
cat >> /etc/modprobe.d/lockd.conf <<EOM
options lockd nlm udpport=32769 nlm tcpport=32803
EOM
```

E reinicie o servidor NFS:

```
service nfs-kernel-server restart
```

Os diretório que serão montados precisam ser criados nos clientes, nessa instalação o mesmo servidor será usado como cliente e servidor.

```
mkdir -p /mnt/primary /mnt/secondary
```

No arquivo /etc/fstab é configurado as montagens de volumes NFS do lado do cliente, como nesse exemplo:

```
cat >>/etc/fstab <<EOM
IP=11.11.11.30:/export/primary /mnt/primary nfs
rsize=8192,wsize=8192,timeo=14,intr,vers=3,noauto 0 2
IP=11.11.11.30:/export/secondary /mnt/secondary nfs
rsize=8192,wsize=8192,timeo=14,intr,vers=3,noauto 0 2
EOM
```

Agora os diretórios exportados podem ser montados com os comandos:

```
mount /mnt/primary
mount /mnt/secondary
```

Para a nuvem CloudStack funcionar, no mínimo 3 VMs do sistema precisam ser executadas, elas podem ser baixadas pois são desenvolvidas pela comunidade de desenvolvedores da ferramenta. Podem ser baixadas e colocadas no diretório apropriado com o comando:

```
/usr/share/cloudstack-common/scripts/storage/secondary/cloud-install-sys-templ \
-m /mnt/secondary -u
http://cloudstack.apr-get.eu/systemvm/4.4/systemvm64template-4.4.1-7-kvm.qcow2.bz2 -h kvm
-F
```

É recomendado ainda que sejam aberta no firewall algumas portas que podem ser usadas pela nuvem CloudStack e caso estejam fechadas impedirão o funcionamento dos serviços da nuvem:

```
ufw allow proto tcp from any to any port 22
ufw allow proto tcp from any to any port 1798
ufw allow proto tcp from any to any port 16509
ufw allow proto tcp from any to any port 5900:6100
ufw allow proto tcp from any to any port 49152:49216
```

Instalação de agente CloudStack

Nesse servidor é onde as máquinas virtuais serão executados, a alocação é controlada pelo gerente de nuvem. Os primeiros passos são instalar pacotes essenciais e configurar repositórios e interfaces de rede.

Instale o servidor NTP para manter os horários atualizados:

```
apt-get install openntp
```

O próximo passo é configurar a rede no servidor, a ferramenta CloudStack usa as bridges linux para criar interfaces virtuais para as instâncias. Instale os pacotes necessários com o comando:

```
apt-get install bridge-utils
```

Faça ainda uma cópia do arquivo de interfaces, pois ele será modificado:

```
cp /etc/network/interfaces /etc/network/interfaces.orig
```

O comando abaixo fixa o IP 11.11.11.31 ao servidor, configura a máscara, gateway e bridge cloudbr0 ligada a interface física eth0.

```
cat >/etc/network/interfaces <<EOM
```

```
auto lo
```

```
iface lo inet loopback
```

```
auto eth0
```

```
iface eth0 inet manual
```

```
# Public network
```

```
auto cloudbr0
```

```
iface cloudbr0 inet static
```

```
address 11.11.11.31
```

```
netmask 255.255.255.0
```

```
gateway 11.11.11.20
```

```
dns-nameservers 8.8.8.8
```

```
bridge_ports eth0
```

```
bridge fd 5
```

```
bridge stp off
```

```
bridge maxwait 1
```

```
# Private network
```

```
auto cloudbr1
```

```
iface cloudbr1 inet manual
```

```
bridge_ports none
```

```
bridge fd 5
```

```
bridge stp off
```

```
bridge maxwait 1
```

```
EOM
```

Agora reinicie o sistema operacional para aplicar as mudanças nas interfaces.

Depois é possível iniciar a instalação dos pacotes CloudStack, primeiro adicionado o repositório, as chaves APT e atualizando o sistema:

```
cat >/etc/apt/sources.list.d/cloudstack.list <<EOM
deb http://cloudstack.apt-get.eu/ubuntu trusty 4.4
EOM
wget -O - http://cloudstack.apt-get.eu/release.asc|apt-key add -
apt-get update
```

É necessário ainda instalar o cliente NFS, que faz a montagem dos diretórios:

```
apt-get install nfs-common
```

Configure a inicialização e portas usadas pelo serviço NFS:

```
cp /etc/default/nfs-common /etc/default/nfs-common.orig
sed -i 'NEED_STATD=/ a NEED_STATD=yes' /etc/default/nfs-common
sed -i '/STATDOPTS=/ a STATDOPTS="--port 662 --outgoing-port 2020"
/etc/default/nfs-common
diff -du /etc/default/nfs-common.orig /etc/default/nfs-common
```

Configure o lockd:

```
cat >> /etc/modprobe.d/lockd.conf <<EOM
options lockd nlm udpport=32769 nlm tcpport=32803
EOM
```

E reinicie o servidor NFS:

```
service nfs-kernel-server restart
```

Os diretório que serão montados precisam ser criados nos clientes, nessa instalação o mesmo servidor será usado como cliente e servidor.

```
mkdir -p /mnt/primary /mnt/secondary
```

No arquivo /etc/fstab é configurado as montagens de volumes NFS do lado do cliente, como nesse exemplo:

```
cat >>/etc/fstab <<EOM
IP=11.11.11.30:/export/primary /mnt/primary nfs
rsize=8192,wsiz=8192,timeo=14,intr,vers=3,noauto 0 2
IP=11.11.11.30:/export/secondary /mnt/secondary nfs
rsize=8192,wsiz=8192,timeo=14,intr,vers=3,noauto 0 2
EOM
```

Agora os diretórios exportados podem ser montados com os comandos:

```
mount /mnt/primary  
mount /mnt/secondary
```

Agora instale o agente do CloudStack

```
apt-get install cloudstack-agent
```

Após é necessária a configuração de parâmetros do libvirt.

```
sed -i '/#listen_tls = 0/ a listen_tls = 0' /etc/libvirt/libvirtd.conf  
sed -i '/#listen_tcp = 1/ a listen_tcp = 1' /etc/libvirt/libvirtd.conf  
sed -i '/#tcp_port = "16509"/ a tcp_port = "16509"' /etc/libvirt/libvirtd.conf  
sed -i '/#auth_tcp = "sasl"/ a auth_tcp = "none"' /etc/libvirt/libvirtd.conf  
diff -du /etc/libvirt/libvirtd.conf.orig /etc/libvirt/libvirtd.conf
```

E também do arquivo libvirt.bin:

```
sed -i -e 's/libvirtd_opts="-d"/libvirtd_opts="-d -l/' /etc/default/libvirt-bin  
diff -du /etc/default/libvirt-bin.orig /etc/default/libvirt-bin
```

Configure ainda o qemu, para o correto funcionamento da VMs.

```
sed -i '/# vnc_listen = "0.0.0.0"/ a vnc_listen = "0.0.0.0"' /etc/libvirt/qemu.conf  
diff -du /etc/libvirt/qemu.conf.orig /etc/libvirt/qemu.conf
```

Desabilite o AppArmor que pode impactar no funcionamento da nuvem:

```
ln -s /etc/apparmor.d/usr.sbin.libvirtd /etc/apparmor.d/disable/  
ln -s /etc/apparmor.d/usr.lib.libvirt.virt-aa-helper /etc/apparmor.d/disable/  
apparmor_parser -R /etc/apparmor.d/usr.sbin.libvirtd  
apparmor_parser -R /etc/apparmor.d/usr.lib.libvirt.virt-aa-helper
```

Reinicie o serviço libvirt para aplicar as mudanças:

```
service libvirt-bin restart
```

E configure as portas dos firewall:

```
ufw allow proto tcp from any to any port 22  
ufw allow proto tcp from any to any port 1798  
ufw allow proto tcp from any to any port 16509  
ufw allow proto tcp from any to any port 5900:6100  
ufw allow proto tcp from any to any port 49152:49216
```

E reinicie o serviço da nuvem:

`service cloudstack-agent restart`

Acessando a interface gráfica a montando a infraestrutura de nuvem

Acesse a infraestrutura que nesse caso sera no endereço <http://11.11.11.30:8080/client> usando o username admin e a senha password.

Selecione a opção "Continue with basic installation". Isso iniciará as configurações para a criação de um ambiente básico de nuvem

- Adicione uma nova zona chamada "zone1", DNS1 8.8.8.8 and Internal DNS 11.11.11.30.
- Adicione o Pod como nome "pod1", o gateway da sua rede 11.11.11.1, mascara de rede 255.255.255.0, IP range 11.11.11.60-11.11.11.99.
- Adicione a guest network, gateway 11.11.11.1, netmask 255.255.255.0, IP range 11.11.11.100-11.11.11.199.
- Adicione um cluster denominado cluster1, Hypervisor KVM.
- Adicione um host, que nesse caso é onde o agente do cloudstack está sendo executado, coloque o IP do nodo, usuário root e sua senha.
- Adicione a zona de armazenamento primária: nome primary1, protocolo NFS, Scope Cluster, server 11.11.11.30, e caminho /export/primary.
- Adicione a zona de armazenamento secundária: NFS server 11.11.11.30, caminho /export/secondary.
- Aperte a opção "Launch" para iniciar a nuvem.

B. Artigos Escritos

Este Capítulo apresenta as referências de todos os artigos publicados durante a realização do projeto HiPerfCloud no ano de 2015.

Vogel, A.; Griebler, D.; Maron, C. A. F.; Schepke C.; Fernandes, L. G.. Private IaaS Clouds: A Comparative Analysis of OpenNebula, CloudStack and OpenStack. 24st Euromicro International Conference on Parallel, Distributed and Network Based Processing (PDP), 2016.

Vogel, A.; Maron, C. A. F.; Griebler, D.; Schepke C.. Medindo o Desempenho de Implantações de OpenStack, CloudStack e OpenNebula em Aplicações Científicas. 16th Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul (ERAD/RS), 2016.

Roveda, D.; Vogel, A.; Griebler, D. Understanding, Discussing and Analyzing the OpenNebula and OpenStack's IaaS Management Layers. Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação (REABTIC), 2015.

Roveda, D.; Vogel, A.; Maron, C. A. F.; Griebler, D.; Schepke, C.. Analisando a Camada de Gerenciamento das Ferramentas CloudStack e OpenStack para Nuvens Privadas. 13th Escola Regional de Redes de Computadores (ERRC), 2015.

Roveda, D.; Vogel, A.; Souza, S.; Griebler, D.. Uma Avaliação Comparativa dos Mecanismos de Segurança nas Ferramentas OpenStack, OpenNebula e CloudStack. Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação (REABTIC), 2015.

Vogel, A.; Maron, C. A. F.; Benedetti, V. L. L.; Shubeita, F.; Schepke C.; Griebler, D. . HiPerfCloud: Um Projeto e Alto Desempenho em Nuvem. 14th Jornada de Pesquisa SETREM (SAPS), 2015.