

UNIVERSIDADE FEDERAL DO ABC
CENTRO DE ENGENHARIA MODELAGEM E CIÊNCIAS SOCIAIS APLICADAS
ENGENHARIA DE INFORMAÇÃO

RAFAEL VINICIUS DE LIMA SOARES

**COMPUTAÇÃO EM NUVEM: IMPLEMENTAÇÃO E TESTES DE DESEMPENHO
DE UMA NUVEM PRIVADA UTILIZANDO OPENSTACK**

SANTO ANDRÉ - SP

2018

RAFAEL VINICIUS DE LIMA SOARES

**COMPUTAÇÃO EM NUVEM: IMPLEMENTAÇÃO E TESTES DE DESEMPENHO
DE UMA NUVEM PRIVADA UTILIZANDO OPENSTACK**

Monografia apresentada como parte dos requisitos para avaliação da disciplina Trabalho de Graduação III do Bacharelado em Engenharia de Informação da Universidade Federal do ABC.

Orientador: Prof. Dr. João Henrique Kleinschmidt

SANTO ANDRÉ – SP

2018

Agradecimentos

Agradeço à Universidade Federal do ABC pela oportunidade de ter acesso à infraestrutura e educação de qualidade durante a minha graduação. Agradeço a todos os professores, técnicos e colaboradores da universidade, em especial ao meu professor orientador Dr. João Henrique Kleinschmidt, pelo conhecimento, ajuda e paciência durante todo o trabalho de graduação.

Agradeço aos meus amigos, de dentro e fora da universidade, mas principalmente aos da Engenharia de Informação que me deram força e compartilharam seus conhecimentos para juntos superarmos as barreiras dessa jornada.

Agradeço à minha família, especialmente a minha mãe, Alecssandra, por me incentivar, dar suporte e ser o meu maior exemplo para encarar os grandes desafios. A minha namorada, Renata, pelo apoio e motivação em todos os momentos. Por fim, agradeço ao meu falecido avô, Aparecido, por sempre ser uma grande inspiração.

Resumo

A computação em nuvem é uma tecnologia que nasceu com o objetivo de oferecer serviços de tecnologia da informação sob demanda e dessa forma reduzir custos e aumentar capacidades computacionais de forma sob demanda e rápida. Os serviços de computação em nuvem estão cada vez mais presentes na vida das pessoas mesmo sem elas perceberem, seja no armazenamento de arquivos ou na utilização de um aplicativo. Este trabalho de graduação faz um estudo teórico sobre computação em nuvem e suas principais características e funcionalidades. Além disso, faz um estudo prático de um ambiente de uma nuvem privada utilizando o Openstack, o qual é um software que auxilia na gestão da nuvem e pode ser utilizado para controlar uma nuvem pública, privada, híbrida ou comunitária. A instalação teve auxílio do Devstack e depois do ambiente configurado foi possível fazer alguns testes de desempenho tanto na visão de usuário como também utilizando o Rally, que é um software que realiza benchmarks para ambientes Openstack.

Palavras-chave: Computação em nuvem, Nuvem privada, Openstack, Devstack, Rally.

Abstract

The cloud computing is a technology that was developed with the aim of offering information technology services on demand, and thus reducing costs and increasing computing capacities in an elastic and rapid way. The cloud computing services have become increasingly present in people's lives even without them realizing it, whether in file storage or in the use of apps. This bachelor dissertation presents a theoretical study about cloud computing and its main aspects and functionalities. In addition, it does a practical study of a private cloud environment, using the Openstack, a software that assists in cloud management and it can be used to control a public, private, hybrid, or community cloud. The installation had assistance of the Devstack and after the environment configured, it was possible to carry out performance testing both on the user view and using the Rally, a software that performs benchmarks for Openstack environments.

Keywords: Cloud computing, Private cloud, Openstack, Devstack, Rally.

SUMÁRIO

1	Introdução	2
1.1	Motivação	2
1.2	Objetivos.....	3
1.3	Estrutura do trabalho	3
2	Computação em nuvem	4
2.1	Conceito.....	5
2.2	Modelos de serviço	8
2.2.1	IaaS.....	9
2.2.2	PaaS	10
2.2.3	SaaS	10
2.2.4	XaaS	10
2.3	Modelos de implantação	10
2.3.1	Nuvem pública	11
2.3.2	Nuvem privada	11
2.3.3	Nuvem híbrida	11
2.3.4	Nuvem comunitária	12
3	Openstack	13
3.1	Conceito.....	13
3.2	Arquitetura	13
3.3	Devstack.....	16
3.4	Rally.....	17
4	Implementação.....	20
4.1	Instalação Openstack utilizando Devstack.....	20
4.2	Análise do ambiente instalado	21
4.3	Criação de usuários/clientes em ambientes distintos	23
5	Testes de desempenho	29
5.1	Teste na visão de usuário.....	30
5.2	Teste utilizando a ferramenta <i>Rally</i>	31
6	Conclusão	38
	Referências	40
	Anexo	42

1 Introdução

1.1 Motivação

Existem diversas definições do que é computação em nuvem, porém uma das mais utilizadas principalmente na área técnica e em certificações de computação em nuvem é a definição do NIST (*National Institute of Standards and Technology*) que diz “Computação em nuvem é um modelo que permite acesso à rede de forma onipresente, conveniente e sob demanda a um conjunto compartilhado de recursos de computação configuráveis que podem ser rapidamente alocados e liberados com o mínimo esforço de gerenciamento ou interação com o prestador de serviço.”[1].

A utilização de computação em nuvem tem potencial de melhorar o acesso a serviços de TI. Dentre essas melhorias destacam-se a democratização da aquisição de recursos, uma vez que os serviços são pagos pelo uso, possibilitando que usuários com pouco ou muito poder financeiro tenham acesso a capacidades semelhantes pagando de acordo com a quantidade que faz dos recursos provisionados. Além disso, o rápido acesso aos serviços traz grandes benefícios para o desenvolvimento e testes de novos projetos, uma vez que a maioria dos serviços são provisionados em minutos ou horas, trazendo grande agilidade se comparado ao modelo tradicional *on premise* que pode levar meses [2].

Os principais provedores dos serviços de computação em nuvem são grandes empresas multinacionais que concorrem num mercado cada mais competitivo, principalmente de nuvem pública, dentre as principais é possível citar AWS, IBM, Microsoft e Google [2]. No entanto, é possível criar um ambiente de computação em nuvem com poucos recursos e utilizando de ferramentas de software aberto, o que é muito comum em pequenos provedores e em nuvens privadas. O Openstack é um software que auxilia na gestão da nuvem e permite criar um ambiente independente do seu tamanho [3].

1.2 Objetivos

O objetivo desse trabalho é estudar o que é computação em nuvem e suas características. A partir de um ambiente real, analisar as suas funcionalidades e alguns dos seus comportamentos.

Para atingir o objetivo foram definidos objetivos específicos:

- Criar um ambiente de nuvem privada com tecnologia Openstack utilizando um computador tradicional.
- Realizar testes de desempenho do ambiente configurado na visão do usuário e utilizando um software específico de benchmarks.

1.3 Estrutura do trabalho

O trabalho foi dividido em 6 capítulos, sendo o primeiro esse com introdução e detalhamento da motivação, objetivo e estrutura do trabalho. No capítulo 2 é abordado o tema de computação em nuvem, uma visão geral com o conceito, modelos de serviço e modelos de implantação. No capítulo 3 o foco é o Openstack, foi explicado desde o conceito, passando pela arquitetura. O Devstack e o Rally que são ferramentas que auxiliam na instalação e testes de desenvolvimento respectivamente também são abordadas no capítulo 3. No capítulo 4 é mostrado passo a passo a implementação do ambiente desde a instalação, configuração e funcionalidades. Já no capítulo 5 são realizados testes de desempenho na visão do usuário da nuvem e também através do Rally. Por fim, no capítulo 6 são apresentadas as conclusões acerca do trabalho desenvolvido. Há no final do trabalho anexos com códigos e explicações de algumas etapas da parte prática.

2 Computação em nuvem

Ao longo dos anos o modelo de arquitetura de TI mudou bastante, pelo modelo de negócio e principalmente pela tecnologia de cada época. No início dos anos de 1970 os *mainframes* surgiram como grande revolução para as empresas, a computação era centralizada, porém existia um alto custo do hardware e do software. A partir dos anos de 1990 outra arquitetura TI surgiu, a cliente/servidor que por ser distribuída baixava o custo do hardware, porém é caracterizada por licenças perpétuas de SO e aplicativos. Nos anos de 2010 a computação em nuvem cresceu e se popularizou, com a tecnologia de grandes Datacenters foi possível baixar os custos e assim otimizar para maior eficiência e agilidade para o pagamento pelo uso [4].

Apesar dos modelos de arquitetura acompanharem a tecnologia de cada época eles não são totalmente substituíveis. Atualmente empresas utilizam desses três modelos diferentes, cada um possui características específicas e trazem benefícios de acordo com o modelo adotado. Algumas empresas utilizam mais de um modelo para as suas operações de acordo com a necessidade e característica da função desejada. Essas diferenças podem ser melhor comparadas na Tabela 1 que descreve as características relacionadas à tecnologia, economia e modelos de negócio das arquiteturas de TI *Mainframe*, *Cliente/Servidor*, *SOA (Software Oriented Architecture)* e *Computação em nuvem* e na Figura 1 que ilustra a evolução da computação em relação ao tempo [5].

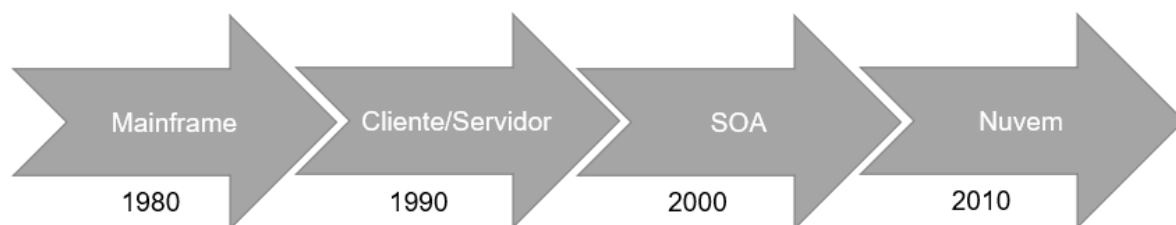


Figura 1 - Evolução da computação pelo tempo.

Tabela 1 - Comparação entre as arquiteturas de TI [5].

	Tecnologia	Economia	Modelo de negócio
Mainframe	Computação centralizada	Otimizado para eficiência por causa do alto custo	Alto custo de hardware e software
Cliente/servidor	Computação distribuída	Otimizado para agilidade devido ao baixo custo	Licença perpétua para SO e aplicativos
Computação em nuvem	Grandes datacenters	Otimizado para eficiência e agilidade	Paga pelo uso

O avanço da banda larga e principalmente o mundo mais dinâmico e ágil para as transformações digitais, também contribuíram para o desenvolvimento da computação em nuvem. Entretanto, a ideia de TI como serviço não é nova e foi apresentada pela primeira vez em 1961 por John McCarthy num evento no MIT (*Massachusetts Institute of Technology*), no qual ele menciona que no futuro a TI poderia ser adquirida como água e energia. Já o termo “computação em nuvem” foi utilizado pela primeira vez em 2005 por Eric Schmidt, CEO do Google. No ano seguinte, a Amazon iniciou o seu serviço de armazenamento, o primeiro serviço de nuvem pública [2].

2.1 Conceito

Baseado na definição de computação em nuvem o NIST [1] apresenta cinco características essenciais para o modelo de computação em nuvem:

- Autosserviço sob demanda
- Amplo acesso à rede
- Conjunto de recursos
- Rápida elasticidade
- Serviços mensuráveis

Autosserviço sob demanda são funcionalidades computacionais capazes de automatizar o processo sem interação humana com o provedor de serviços.

Ampla acesso à rede são os recursos computacionais estarem ao acesso do usuário de forma fácil, seja por meio de uma plataforma ou interface web que possa ser acessada pela rede. No caso de uma nuvem privada por uma rede interna ou no caso de uma nuvem pública por meio da internet.

Conjunto de recursos são os recursos computacionais que podem ser virtuais ou físicos do provedor para servir múltiplos usuários no qual são alocados e realocados dinamicamente conforme a demanda.

Rápida elasticidade são recursos computacionais providos elasticamente e rapidamente liberados. O usuário deve ter a possibilidade de adquirir qualquer quantidade a qualquer momento como recursos ilimitados.

Serviços mensuráveis são capacidades de os sistemas de gerenciamento da nuvem monitorar automaticamente cada tipo de serviço que são armazenamento, processamento e largura de banda. Esse monitoramento deve ser transparente para o usuário e para o provedor de serviços.

Há outros modelos, fundamentalmente diferentes, que usualmente são confundidos com computação em nuvem que são *hosting* e *collocation*.

Hosting é um serviço de aluguel de recursos computacionais no qual o usuário paga o serviço para uma empresa que tem a responsabilidade de dar manutenção num ambiente. No entanto não há serviço sob demanda uma vez que o aluguel é feito baseado num ambiente, tampouco se tem um ambiente elástico, mesmo podendo adquirir novos equipamento o crescimento do ambiente não é trivial. Já o *Collocation* é quando o usuário possui um ambiente, mas não quer se responsabilizar pela manutenção e gastos como eletricidade, climatização e conexão. Nesse caso uma empresa terceira assume essas atividades e faz a cobrança pelos serviços prestados, mas possui praticamente as mesmas limitações do modelo *Hosting* [5].

Essas características fazem a computação em nuvem ter grandes benefícios se comparado com o modelo tradicional *on premise* principalmente pelo autosserviço sob demanda como pode ser visto na Figura 2. Em azul está a demanda do usuário, em cinza o modelo tradicional *on premise* que sempre deve estar acima da demanda gerando ociosidade de recursos, já em laranja está o modelo da nuvem, no qual o

consumo acompanha a demanda sem deixar lacunas de recursos e pagando apenas pelo uso sem ociosidade de recursos.

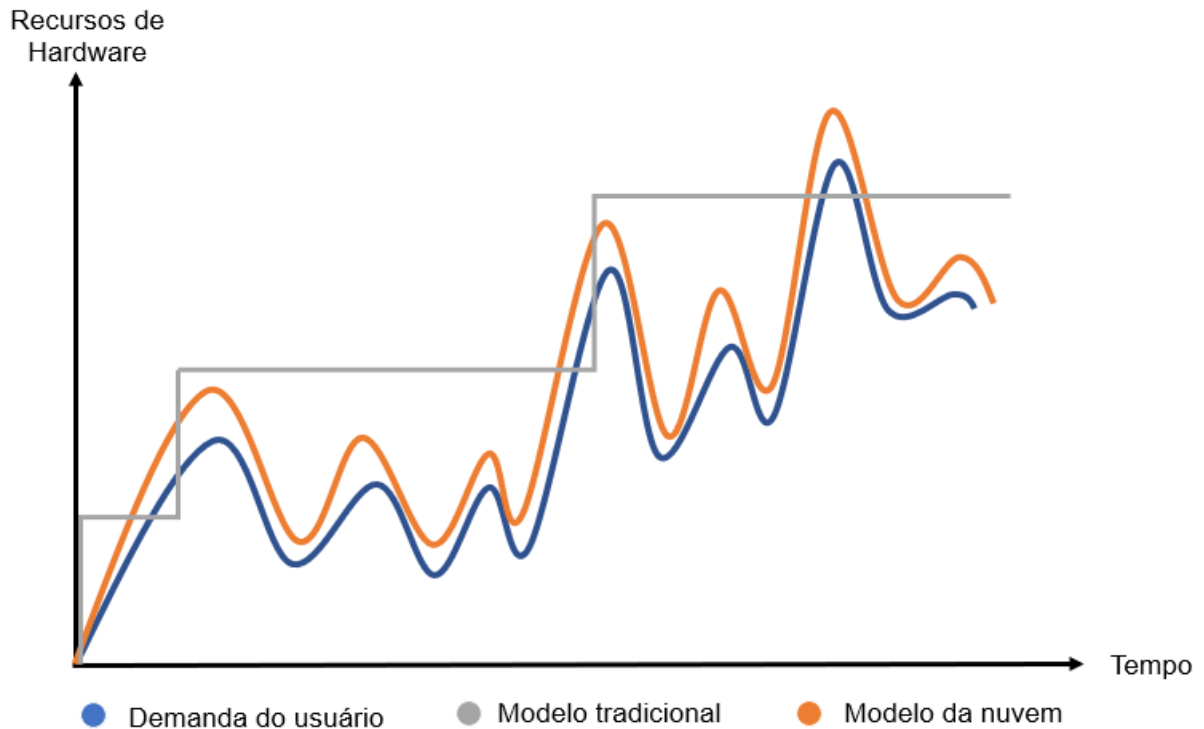


Figura 2 - Diferença computação em nuvem x modelo tradicional.

A computação em nuvem utiliza uma arquitetura diferente dos modelos *mainframe* e cliente/servidor, mas aproveita do benefício de ambos, a utilização de grandes *datacenters* otimizados para eficiência e agilidade. É possível classificar a computação em nuvem por características de modelos de serviço e de implantação. Os modelos de serviço são Infraestrutura como serviço (IaaS), Plataforma como serviço (PaaS) e Software como serviço (SaaS), já os modelos de implantação são nuvem pública, privada, híbrida e comunitária [5].

2.2 Modelos de serviço

Dentre os modelos de serviços fornecidos eles podem ser divididos em três grupos que são IaaS (*Infrastructure as a Service*), PaaS (*Platform as a Service*) e SaaS (*Software as a Service*). Em cada um deles existe um nível de responsabilidade do usuário e do provedor. A Figura 3 ilustra bem as diferenças de responsabilidade de acordo com o modelo de serviço adquirido desde o modelo tradicional *on premise* passando por IaaS, PaaS e chegando em SaaS. Em branco está o que é de responsabilidade do cliente e em preto o que é responsabilidade do provedor.

Além disso, há diferentes interlocutores nos modelos de serviço. Apesar de o provedor fornecer todos os serviços de IaaS, PaaS e SaaS quem irá consumir são usuários diferentes. No modelo IaaS o usuário é um desenvolvedor ou normalmente um responsável pela infraestrutura de uma empresa. No modelo PaaS o desenvolvedor que consome a plataforma para criar e executar projetos. Porém no modelo SaaS o usuário é normalmente o usuário final que só irá consumir o serviço [5]. Essa diferença está ilustrada na Figura 4.

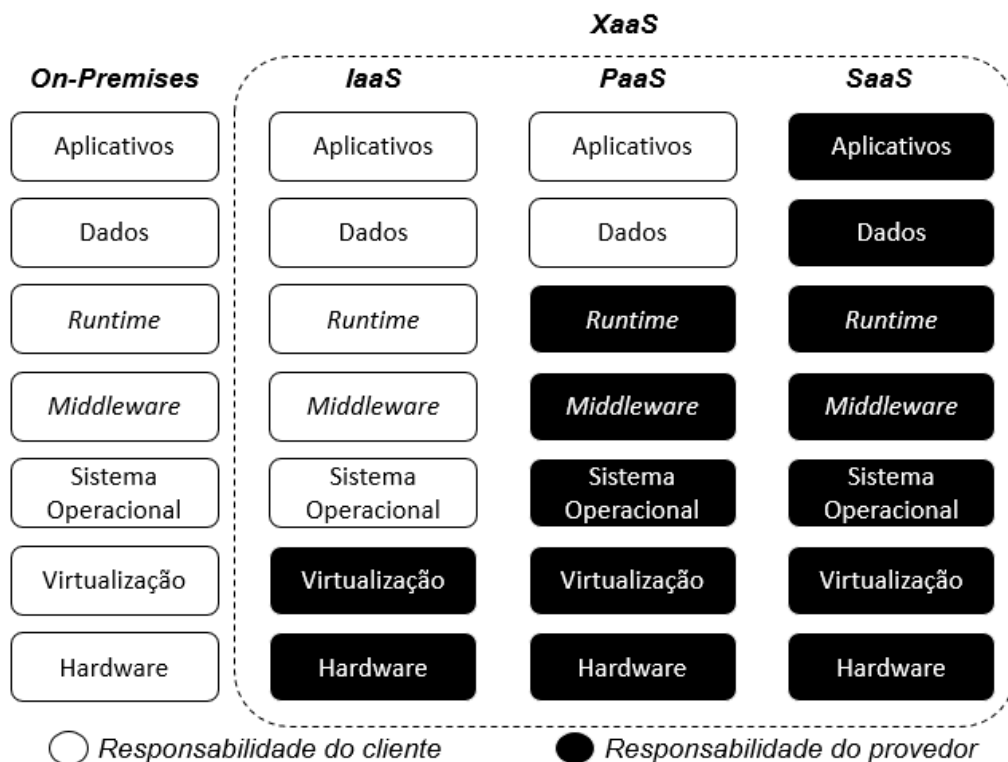


Figura 3 - Comparativo de responsabilidades entre os modelos de serviços.

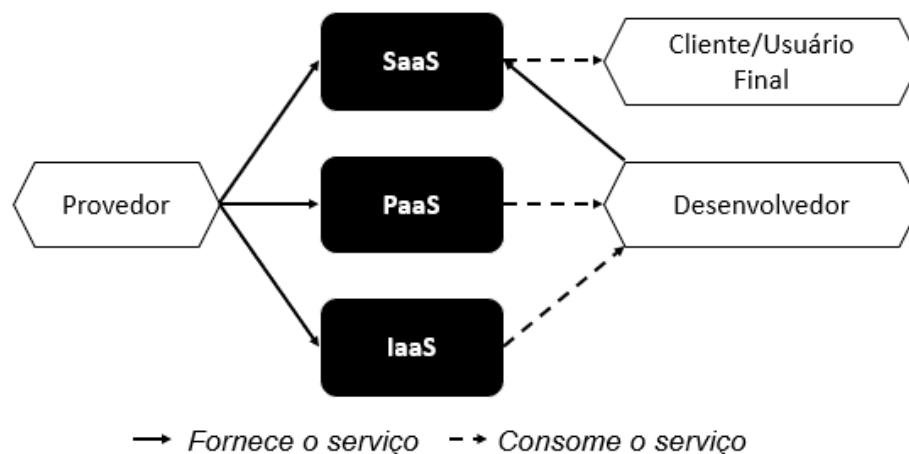


Figura 4 - Papéis em computação em nuvem.

2.2.1 IaaS

O modelo de infraestrutura como serviço, mais conhecido como IaaS, é a oferta que o provedor fornece ao cliente/usuário recursos computacionais, ou seja, o hardware, como servidores, redes e discos de armazenamento. O pagamento normalmente é feito pelo uso (*pay as you go*) pelas máquinas virtuais ou servidores *bare metal* e possui as principais características essenciais da definição de computação em nuvem, principalmente a escalabilidade por qualquer um dos recursos de hardware, seja por unidade de processamento, disco ou memória [5] [6].

O serviço de IaaS é voltado para levar infraestrutura para o cliente/usuário tendo como benefícios o pagamento pelo uso, personalização de ambientes e principalmente a agilidade no provisionamento do hardware que normalmente é de algumas horas. Um mesmo ambiente *on-premises* pode levar meses desde a compra do hardware até a instalação [7].

2.2.2 PaaS

O modelo de plataforma como serviço, mais conhecido como PaaS, é um serviço voltado para o desenvolvimento. É uma oferta que o provedor fornece um ambiente em nuvem para desenvolver, administrar e entregar aplicações. Nesse ambiente o cliente/usuário não precisa se preocupar com a infraestrutura e sim apenas com o banco de dados e a aplicação.

É fornecida ao consumidor a capacidade para implantar sobre a infraestrutura de nuvem aplicações criadas ou adquiridas usando linguagens de programação, bibliotecas, serviços e ferramentas suportadas pelo provedor [5] [6].

2.2.3 SaaS

O modelo de software como serviço, mais conhecido como SaaS, é uma oferta voltada diretamente para o usuário final. Trata-se de um aplicativo hospedado na nuvem que pode ser acessado pelos clientes por meio da rede. Nesse ambiente o usuário não precisa se preocupar com a infraestrutura nem com o desenvolvimento do aplicativo, ele apenas o consome [5] [6].

2.2.4 XaaS

O modelo XaaS tem significado de qualquer coisa como serviço e engloba os modelos IaaS, PaaS e SaaS. Porém, se refere ao crescimento dos tipos de serviços que podem ser oferecidos pela rede no modelo de computação em nuvem [6].

2.3 Modelos de implantação

Dentre os modelos de implantação há os modelos de nuvem pública, privada híbrida e comunitária.

2.3.1 Nuvem pública

Nuvem pública é um modelo de compartilhamento de recursos computacionais. Esse é o modelo mais popular e mais acessível. Os provedores oferecem normalmente o serviço pago por hora ou por mês. A segurança e o desempenho de alto nível são problemas enfrentados na nuvem pública [5].

2.3.2 Nuvem privada

O modelo de nuvem privada é o modelo em que não há compartilhamento de recursos. Logo, a ideia de privado não está ligada a localização física do ambiente computacional, ele pode estar no endereço do cliente ou do provedor, mas obrigatoriamente será um recurso dedicado a um único cliente. Porém, mesmo com exclusividade de recursos a nuvem privada possui benefícios similares ao da nuvem pública. A nuvem privada é indicada para usuários que possuem e gerenciam dados sensíveis, um bom exemplo são transações bancárias e financeiras [5].

2.3.3 Nuvem híbrida

O modelo de nuvem híbrida é um conjunto de nuvem pública e nuvem privada. É um modelo crescente no qual o cliente aproveita o melhor dos dois modelos como o baixo custo e escalabilidade da nuvem pública e a segurança privacidade e controle total da nuvem privada como ilustra a Figura 5. Também é uma realidade da maioria dos usuários que já possuem parte da infraestrutura *on-premisse* e querem aproveitar os benefícios da nuvem, isso faz com que a nuvem híbrida seja um modelo bastante adotado [7].

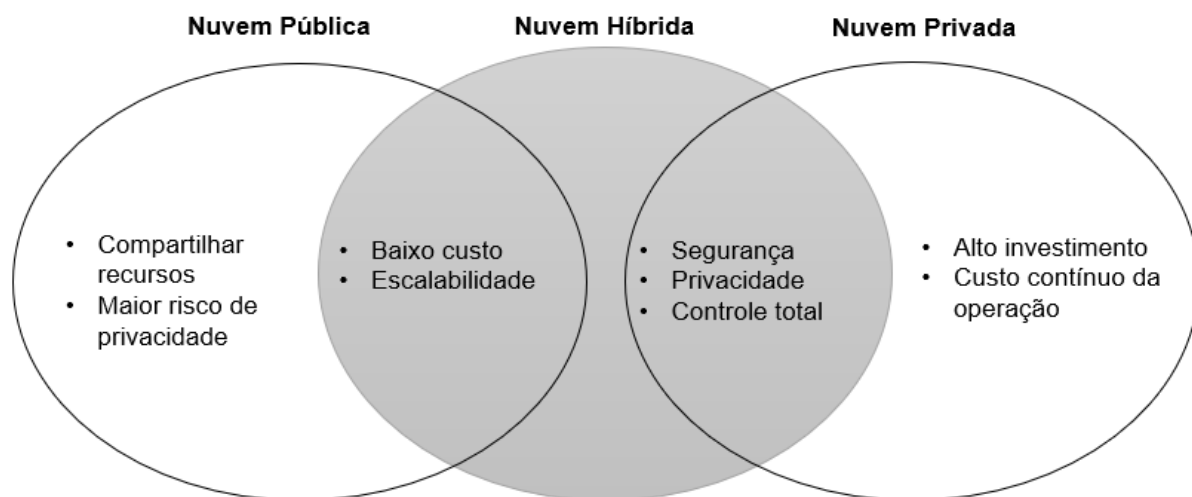


Figura 5 - Modelo nuvem híbrida.

2.3.4 Nuvem comunitária

A nuvem comunitária é semelhante a nuvem pública, porém os usuários são controlados e restritos. Ou seja, diferentes pessoas ou instituições que possuem um objetivo em comum contratam a infraestrutura de uma nuvem privada para uso comunitário [5].

A nuvem comunitária é comum em centros de pesquisa, universidades e pequenas empresas com interesse comum. Há nesse caso um benefício tecnológico e financeiro para os participantes que adotam esse modelo [8].

3 Openstack

3.1 Conceito

O Openstack é uma plataforma de código aberto que pode ser utilizada para o gerenciamento de nuvens públicas e privadas que foi desenvolvida por uma colaboração global de desenvolvedores, entre eles a NASA e Rackspace. Trata-se de uma tecnologia que possui diversos projetos para diferentes componentes e que juntos constituem uma solução de infraestrutura em nuvem. O Openstack é de simples implementação e possui todas as características de um serviço de computação em nuvem [3] [9].

Openstack é o sistema operacional da nuvem que automatiza, controla e gerencia amplos serviços de computação, armazenamento e recursos de rede. O acesso a essas informações é dado por meio de uma interface web chamada Horizon, a qual permite que os usuários possam liberar ou requisitar recursos de acordo com suas necessidades e os administradores possam controlar os recursos conectados ao Openstack [10].

3.2 Arquitetura

A arquitetura básica do Openstack é um conjunto de recursos computacionais interconectados numa rede comum que podem ser físicos ou virtuais, como por exemplo *Bare Metals*, máquinas virtuais, *containers* e equipamentos de armazenamento como *object storage*, *file storage* e *block storage* [10]. Todas as aplicações trabalham nesse ambiente e podem ser acessadas via rede. Além disso, APIs de *dashboard* e de monitoramento suportam as aplicações que estão nesse ambiente e principalmente permitem o gerenciamento da nuvem. A Figura 6 ilustra em detalhes essa visão geral do ambiente Openstack.

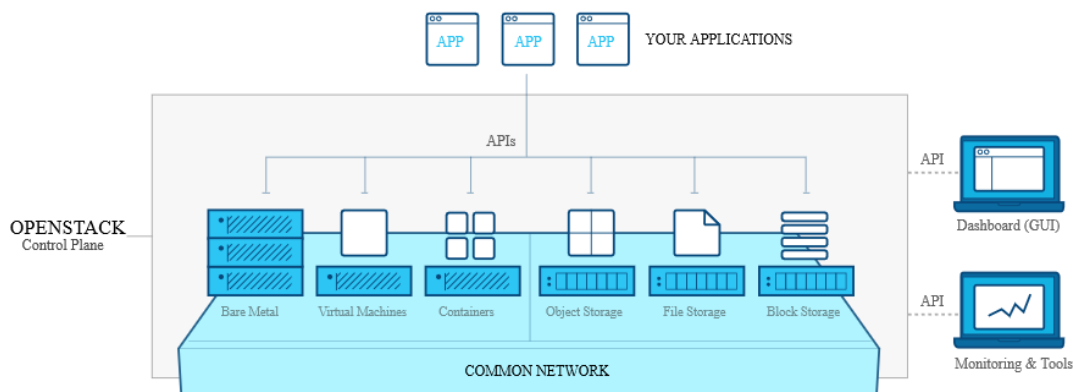


Figura 6 - Visão geral Openstack [10].

Dessa forma, o Openstack é dividido em serviços básicos e essenciais para o funcionamento da nuvem. Alguns serviços são opcionais e ficam a critério do provedor. De acordo com [11], os serviços são:

- **Horizon:** Esse módulo permite que administradores e usuários acessem os recursos por meio de um *dashboard* numa interface web e consigam executar diversas ações como visualizar, criar, excluir, e gerenciar máquinas virtuais, rede, monitoramento de status e eventos, fornecendo maiores capacidades de solução de problemas.
- **Nova:** Esse módulo tem a responsabilidade de fazer a administração dos *hypervisors*, além de gerenciar todo o ciclo de vida das instâncias.
- **Swift:** Esse módulo é utilizado para armazenar dados não estruturados. Ele consegue guardar e recuperar grandes quantidades de objetos. Seu objetivo é escalar e otimizar a durabilidade e simultaneidade em todo o conjunto de dados.
- **Cinder:** Esse módulo facilita na hora de criar e administrar dispositivos de armazenamento de blocos. Em outras palavras, o Cinder possibilita ao usuário utilizar volumes adicionais para sua estrutura.
- **Glance:** Esse módulo é responsável pelas imagens dos sistemas operacionais presentes no Openstack.
- **Keystone:** Esse módulo cuida de toda a parte de autenticação de serviços e usuários. Em outras palavras, o Keystone autoriza que um módulo do

Openstack consiga se comunicar com outros, além de gerenciar o que cada usuário pode fazer dentro da nuvem.

- **Neutron:** Esse módulo inclui uma série de APIs, plug-ins e softwares que, basicamente, garante a transparência na comunicação entre dispositivos e tecnologias dentro de ambientes IaaS.

O administrador do ambiente Openstack deve entender a arquitetura lógica de todo o sistema para adequar melhor o uso de cada recurso e como eles interagem e se relacionam. Na Figura 7 é ilustrada uma arquitetura geral dos projetos Openstack que são *Openstack Dashboard*, *Openstack Object Storage*, *Openstack Image Service*, *Openstack Compute*, *Openstack Block Storage*, *Openstack Networking* e *Openstack Identity Service*. Cada um desses projetos possui sua arquitetura específica que envolve serviços e banco de dados que se relacionam entre si e entre os outros projetos. Como pode ser visto, todos os projetos possuem acesso direto com a internet e podem ser acessados via API ou HTTP(S). As setas vermelhas pontilhadas indicam a relação entre projetos diferentes. É importante notar que o projeto *Openstack dashboard*, que é o Horizon, está conectado com todos os projetos, uma vez que por meio dele é possível fazer alterações no ambiente da nuvem e controlar as suas funcionalidades de forma fácil. Porém, cada um deles pode ser acessado pela sua respectiva API que está diretamente ligada à rede. Outro ponto importante é que todos os projetos se conectam com o *Openstack Identity Service*, que é o Keystone, ele é responsável pela autenticação de todos os serviços, dessa forma há segurança no acesso de qualquer projeto seja via Horizon ou por API, pois todos são validados pelo Keystone [10].

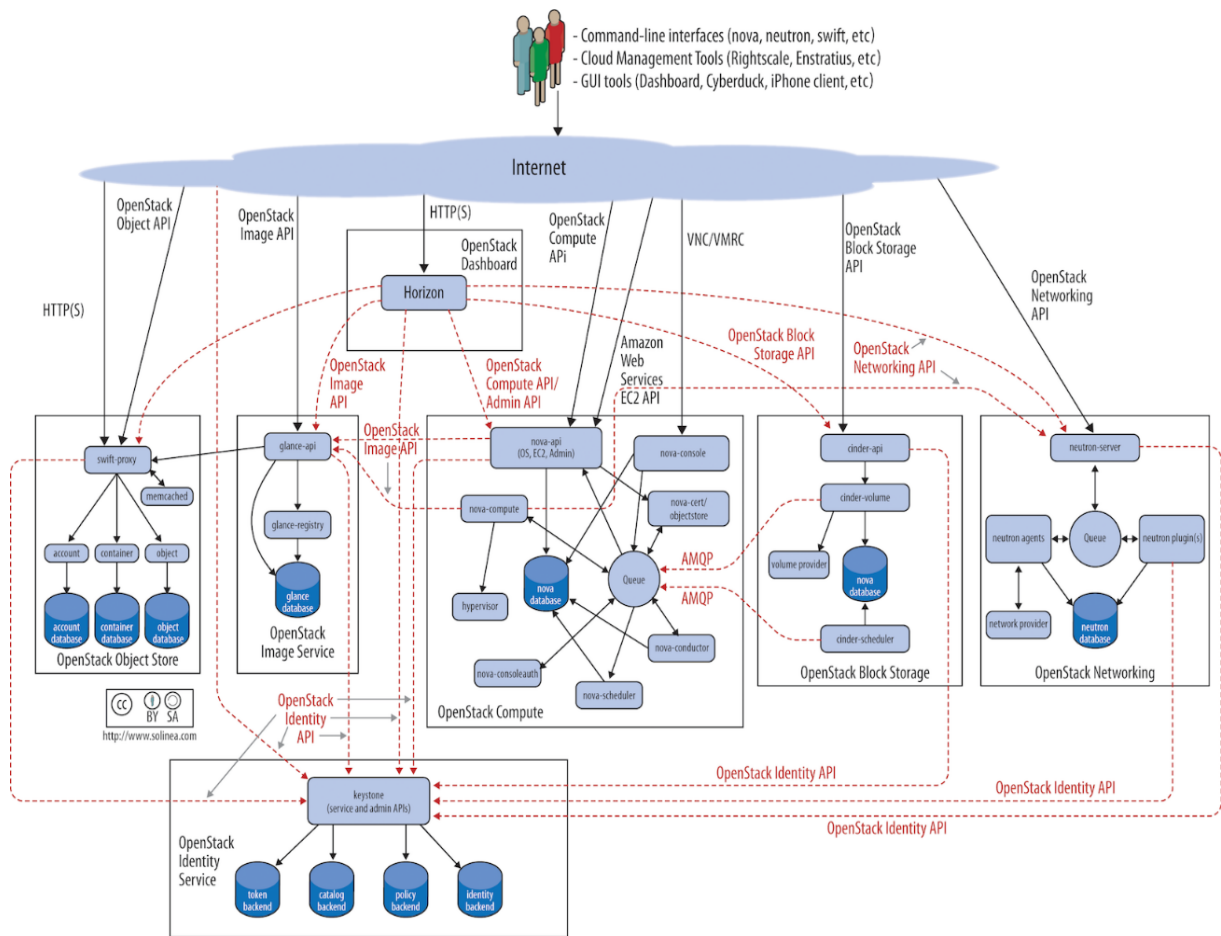


Figura 7 - Arquitetura e integração dos serviços Openstack [10].

3.3 Devstack

Existem diversas formas de fazer a implementação do Openstack, dentre elas a opção de distribuir a instalação por um serviço de cada vez, seja num único hardware ou em hardwares diferentes. Esse procedimento é complexo, porém é o padrão para um ambiente de produção. O Devstack é um roteiro de códigos que automatiza a instalação. Além disso, com o Devstack é possível instalar o Openstack com menos recursos de memória RAM, inclusive pode ser feito em uma máquina virtual Linux, uma vez que é um ambiente voltado para desenvolvimento e testes. Não é aconselhável utilizar o Devstack para um ambiente de produção. Mas como o objetivo do trabalho não foi ter um ambiente em produção esse modelo foi uma boa opção [12].

3.4 Rally

Rally é a uma ferramenta de automação de benchmark capaz de fazer testes em um ambiente Openstack com relação a escalabilidade e verificação de como o ambiente está funcionando como um todo. As principais ações do Rally são fazer a implementação junto ao ambiente Openstack, fazer testes de verificação, benchmarks e gerar relatórios que auxiliam na análise dos testes como pode ser ilustrado na Figura 8 [13]. Todas as ações são feitas num ambiente Openstack, exceto a geração de relatórios que utiliza dos dados armazenadas no banco de dados do Rally. O banco de dados também é responsável pela verificação de resultados. Essa é a arquitetura básica utilizada pelo Rally.

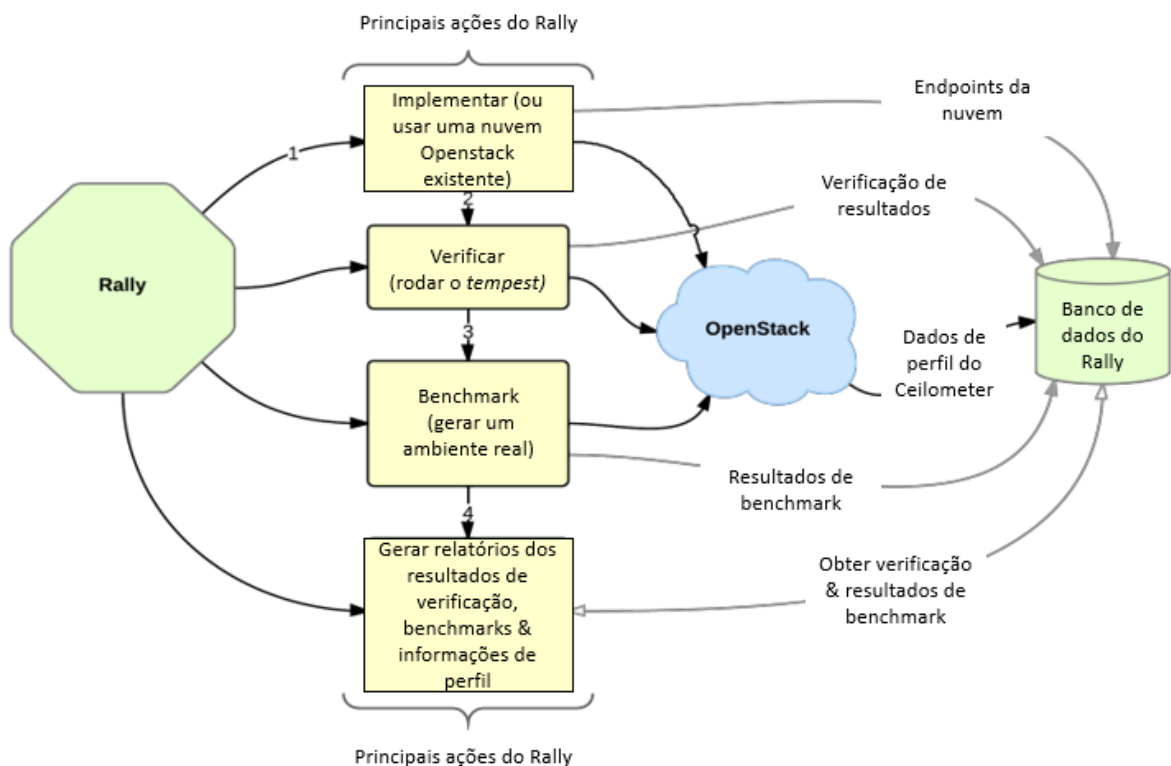


Figura 8 - Funcionalidades do Rally [13].

Os maiores casos de uso do Rally podem ser vistos na Figura 9. O primeiro fluxograma é o de desenvolvimento e testes, o qual é feito a implementação do Rally num ambiente Openstack e são simuladas situações reais, a partir dos resultados obtidos é possível melhorar o sistema ou fazer a implementação num novo ambiente Openstack. Esse processo pode ser contínuo até o ambiente estar apto para as

condições necessárias. Ele tem o papel de testar e melhorar um ambiente. O segundo fluxograma é o de DevOps que é utilizado para desenvolvimento, no qual o Rally é instalado numa nuvem existente e são simuladas situações reais para melhorar os resultados e ter certeza que o ambiente suportará o SLA (*Service Level Agreement*). O último caso é para ambientes de integração e entregas contínuas CI/CD (*Continuous Integration / Continuous Delivery*) no qual o Rally é instalado nas últimas versões de ferramentas ou códigos para realizar benchmarks e avaliá-lo antes de ser feita uma atualização, esse processo é contínuo [14].

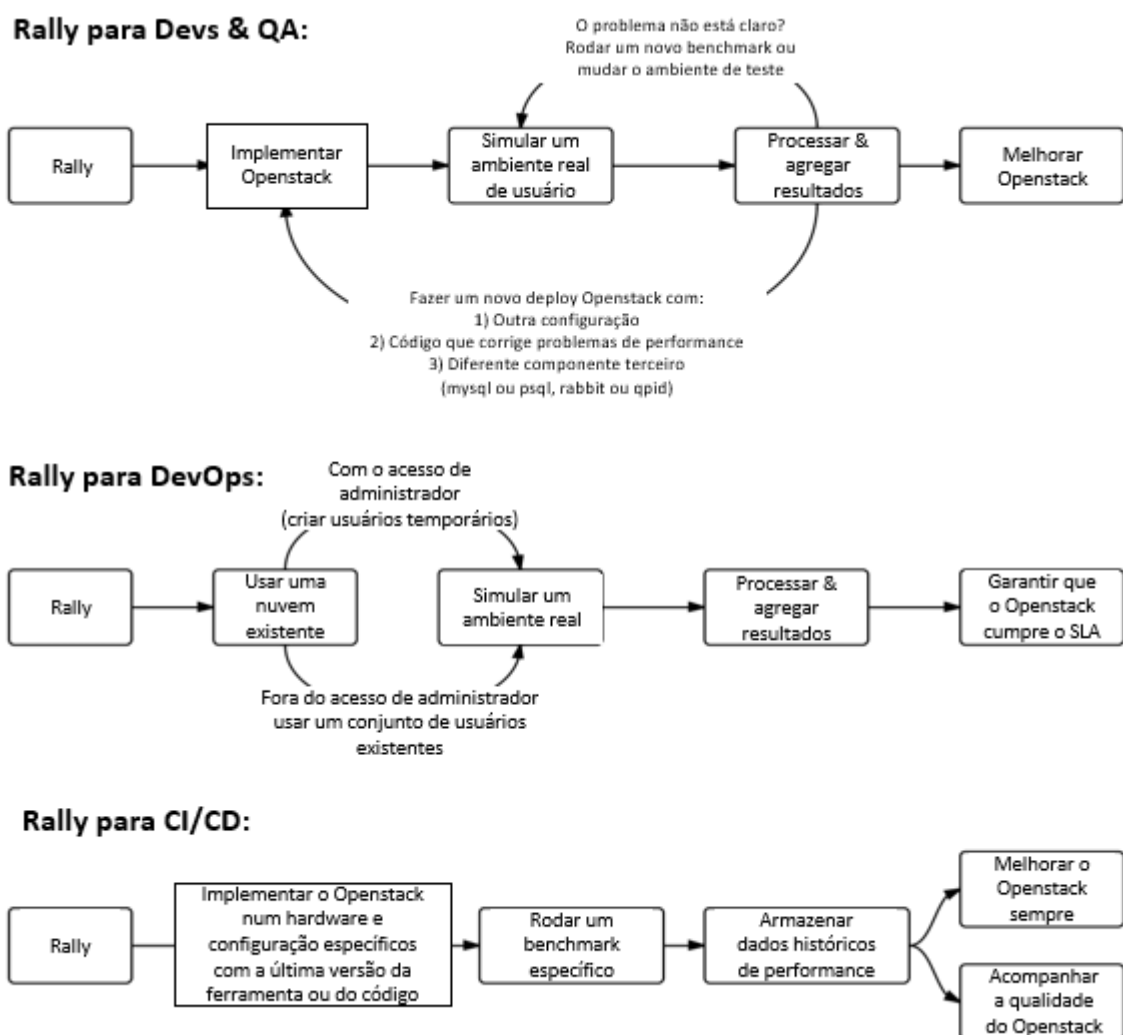


Figura 9 - Casos de uso Rally [14].

Os maiores benefícios do Rally são a automação de medidas para avaliar como novos códigos se comportam no sistema operacional. Detectar problemas de desempenho e variação do ambiente. Investigar como diferentes implementações afetam a performance do sistema operacional. Automatizar a pesquisa para o melhor hardware para o ambiente Openstack. Automatizar a produção de operações de diferentes ambientes [14].

4 Implementação

Primeiramente, foi criada uma máquina virtual utilizando o software Virtualbox. Foi instalado um sistema Linux, nesse caso o Ubuntu 16.04 com processador de 2 vcpu, 10GB de memória RAM e 50GB de disco. Foi necessário colocar um IP fixo na máquina utilizada e criar uma rede interna para que todos os serviços pudessem se comunicar.

4.1 Instalação Openstack utilizando Devstack

Após instalado o sistema operacional Ubuntu 16.04, que suporta o ambiente Openstack [15], numa máquina virtual por meio do Virtualbox. Foi preparado o ambiente para a instalação do Openstack por meio do Devstack. Primeiramente foi criado um usuário 'stack' com privilégios de administrador. A partir desse usuário fez-se a instalação padrão do Devstack conforme pesquisado na comunidade Openstack [16].

O ponto mais crítico da instalação do Devstack é configuração da rede, que se trata especificamente do projeto Neutron que é o responsável pela rede do Openstack. A rede é importante pois todos os serviços são controlados por SDN (*software defined networking*), logo toda a rede incluindo IP's pré configurados devem estar bem definidos para evitar erros na instalação e na utilização do ambiente em si. Devido a essa criticidade as especificações da rede são feitas num arquivo separado que é usado como referência no momento da instalação. Todo o processo e *scripts* utilizados na instalação podem ser vistos em detalhe na seção 'Anexo' no final do trabalho.

Preparado todo o sistema e as configurações de rede o processo de instalação demorou 50 minutos para ser finalizado nas configurações de hardware mencionadas anteriormente. Ao final da instalação foi indicado um acesso ao Horizon para controlar o sistema baseado no host IP.

4.2 Análise do ambiente instalado

Após a instalação do Openstack é possível acessá-lo por meio do host IP pelo Horizon, o qual é uma interface web que permite administrar a nuvem. O acesso é feito por meio de um usuário e senha, como ilustra a Figura 10.

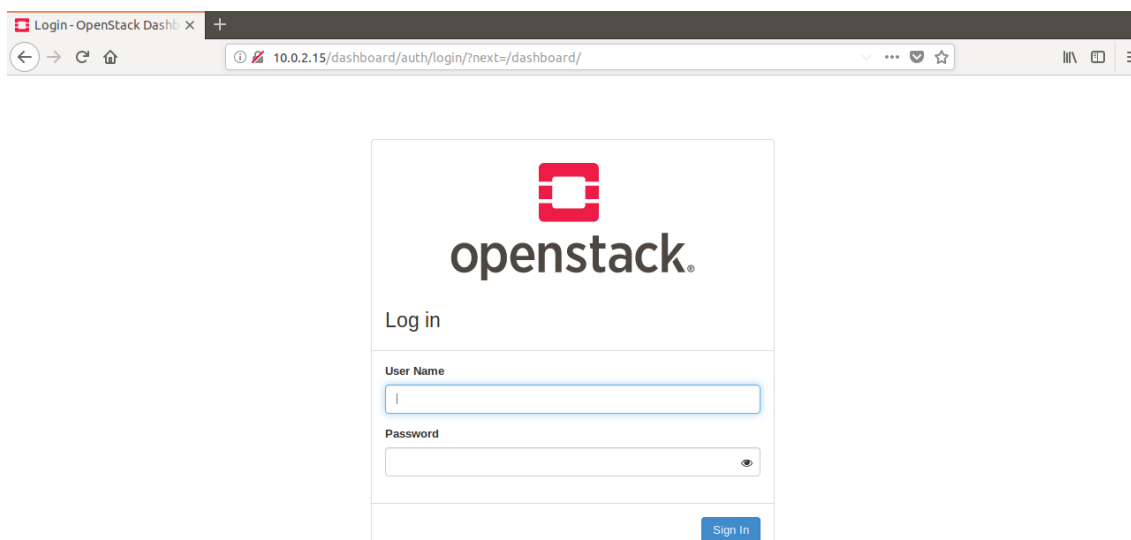


Figura 10 - Acesso ao Openstack via Horizon através do host IP.

Ao acessar o sistema, as configurações são divididas em três tipos que são *Project*, *Admin* e *Identity*, como ilustra a Figura 11, e cada uma tem uma função específica. Na aba *Project* são configurados os ambientes de cada projeto, desde a rede até o tamanho das instâncias e volumes. A aba *Admin* é onde se tem uma visão geral do ambiente Openstack onde estão alocados os recursos, a disponibilidade e saber especificamente a qual projeto e usuário cada um deles está relacionado. Na aba *Identity* são configurados os projetos e usuários, definidas senhas de acesso ao sistema e cotas de uso.

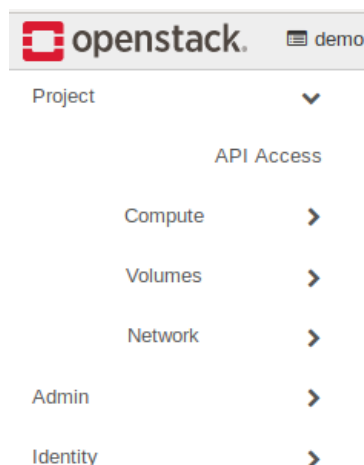


Figura 11 - Tipos de configuração do modo administrador.

Na aba 'Compute' há a opção 'Overview' onde é possível ter um resumo dos recursos que estão sendo utilizados por determinado projeto como ilustra a Figura 12. Esse é um dos princípios da computação em nuvem no qual o usuário e o administrador tem o controle do que está sendo utilizado, tornando transparente a relação entre usuário/cliente e o administrador/provedor.

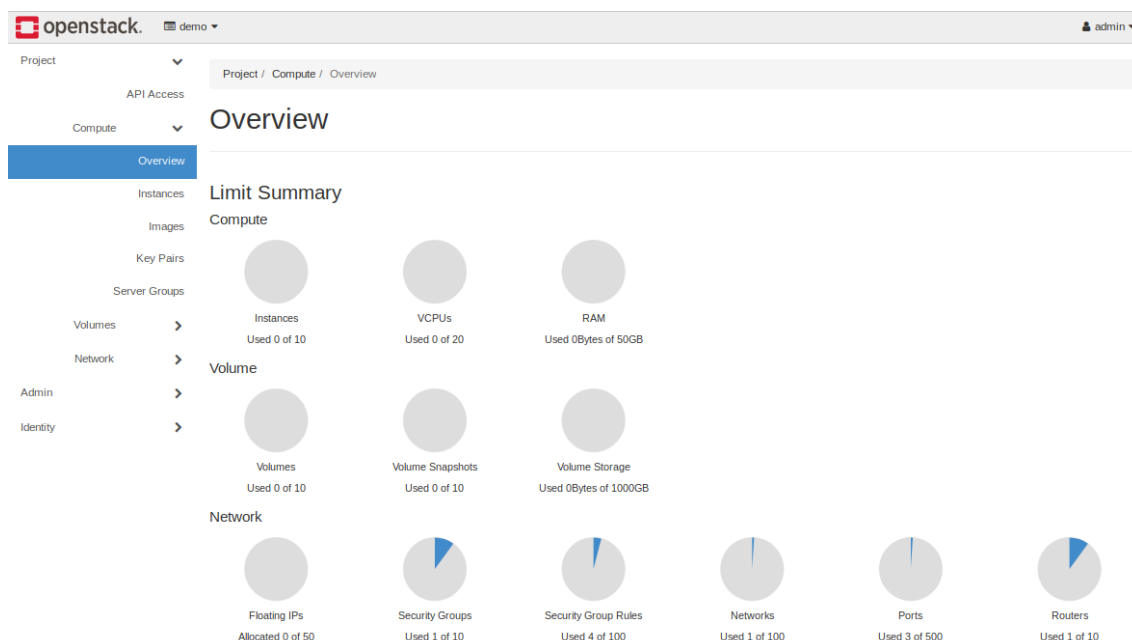


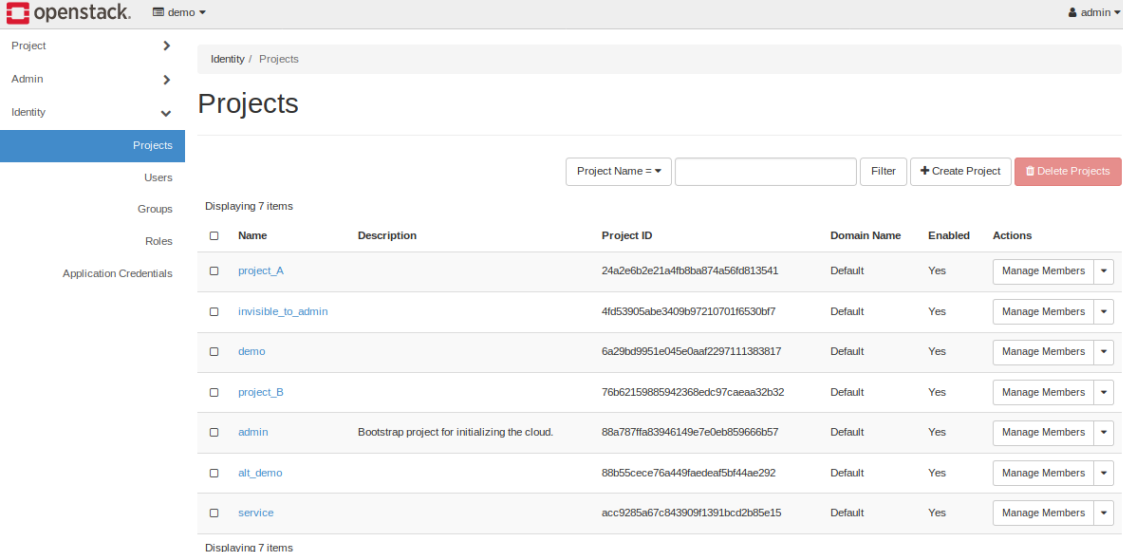
Figura 12 - Visão geral do administrador dos recursos utilizados no sistema.

4.3 Criação de usuários/clientes em ambientes distintos

O compartilhamento de recursos é uma característica fundamental para a eficiência do modelo de computação em nuvem. Todavia, compartilhar recursos não significa perder a privacidade do ambiente. No Openstack é possível que diferentes usuários acessem seus ambientes sem que um interfira no outro. [8] O administrador tem acesso ao todo, porém cada usuário tem acesso apenas ao que lhe foi disponibilizado. O acesso é feito com um login e senha específicos que garantem a privacidade de cada usuário.

Partindo dessa característica do sistema, foram criados dois usuários para testar tal funcionalidade, os usuários 'user_A' e 'user_B'. Esses terão um ambiente simples devido à baixa quantidade de recursos de hardware disponíveis. Trata-se de uma subrede que será tratada como rede interna do usuário, um roteador para conectá-la ao ambiente geral Openstack e uma instância.

Primeiramente são criados os usuários e cadastradas as suas respectivas senhas na aba 'Identity' por meio da conta do administrador. Posteriormente são criados os ambientes, denominados 'projects' no Openstack, cada 'project' é associado a um usuário que foi criado anteriormente e possuem os nomes de 'project_A' e 'project_B' como ilustra a Figura 13. Dessa forma cada usuário terá acesso ao seu respectivo ambiente.



Name	Description	Project ID	Domain Name	Enabled	Actions
project_A		24a2e6b2e21a4fb8ba874a56fd813541	Default	Yes	Manage Members
invisible_to_admin		4fd53905abe3409b97210701f6530b7	Default	Yes	Manage Members
demo		6a29bd9951e045e0aaf2297111383817	Default	Yes	Manage Members
project_B		76b62159885942368edc97caaaa32b32	Default	Yes	Manage Members
admin	Bootstrap project for initializing the cloud.	88a787ffa83946149e7e0eb859666b57	Default	Yes	Manage Members
alt_demo		88b55cece76a449faedeaaf5bf44ae292	Default	Yes	Manage Members
service		acc9285a67c843909f1391bcc2b85e15	Default	Yes	Manage Members

Figura 13 - Projects do Openstack.

Entrando no ambiente por meio do 'user_A' será feita a criação do ambiente com a subrede, roteador e instâncias. O primeiro passo é acessar a aba 'Compute' e posteriormente a aba 'Network' para ser criada a rede interna do usuário. Um roteador é utilizado para fazer a conexão da rede interna com rede pública como mostra a Figura 14. Também são definidos os IP's da rede interna que serão utilizados. Nesse caso a rede interna tem IP's 192.168.0.0/24.

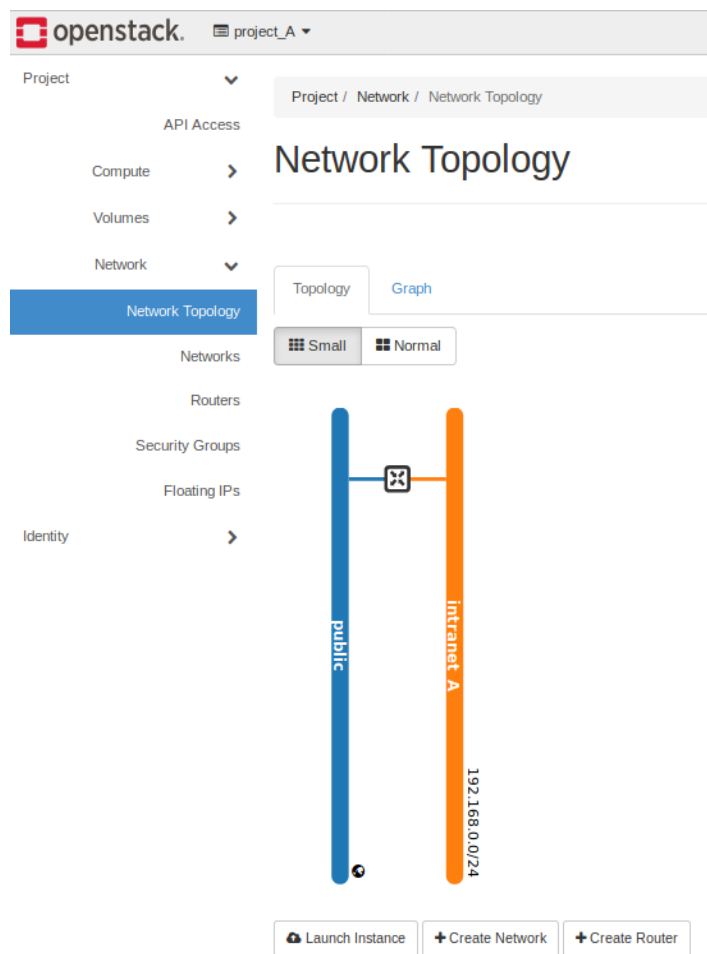


Figura 14 - Criação da rede interna do usuário.

Após a criação da rede interna é possível criar uma instância e associar a ela. Nesse caso foi provisionada uma instância com processador de 1 vcpu, 1GB de RAM e 5GB de disco com uma imagem de um sistema operacional Linux que utiliza pouca memória, processamento e disco, chamado CirrOS como ilustra a Figura 15.

Feito o procedimento da rede interna e o provisionamento da instância é possível analisar a topologia da rede criada pelo user_A. O Openstack oferece duas visões, a topológica que foca nas redes e suas conexões e a gráfica, que cada elemento seja instância, rede, subrede ou roteador são representadas com o mesmo grau de importância, para diferentes casos determinada visão pode ser mais importante para ser analisado como mostra a Figura 16.

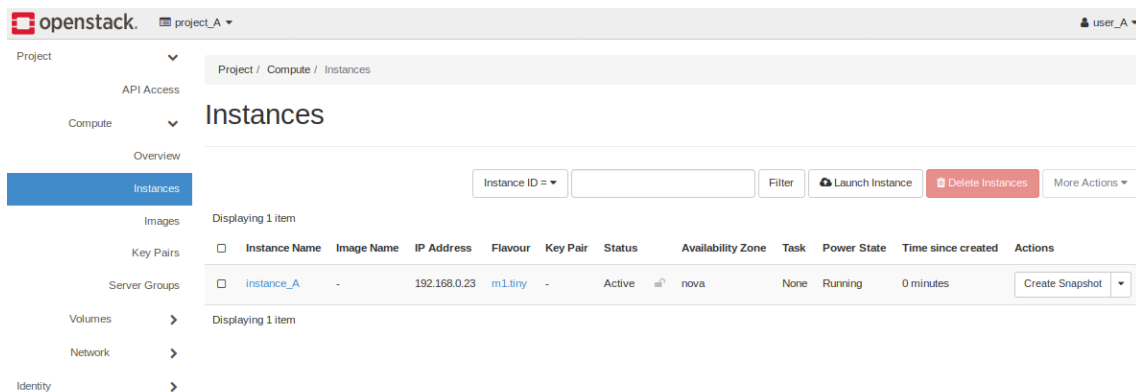


Figura 15 - Criação de uma instância.

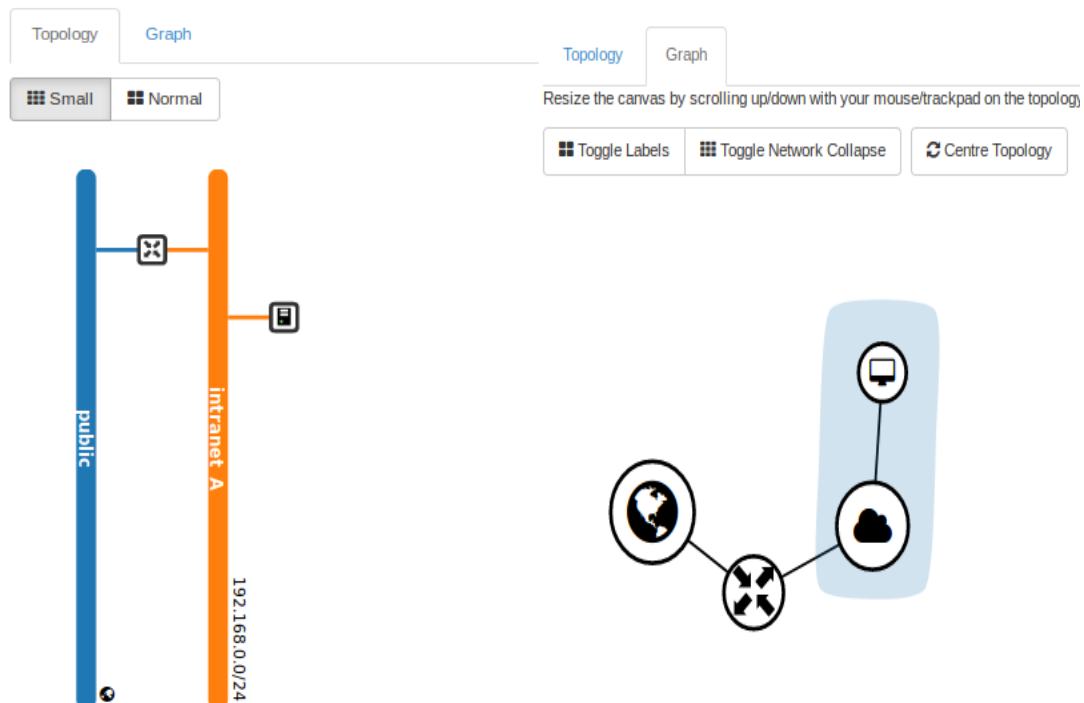


Figura 16 - Topologia do user_A.

A mesma visão dos recursos gerais visto anteriormente para o administrador também oferecida para cada usuário com os recursos que estão sendo utilizados apenas no seu ambiente, como mostra a Figura 17 para o caso do user_A.

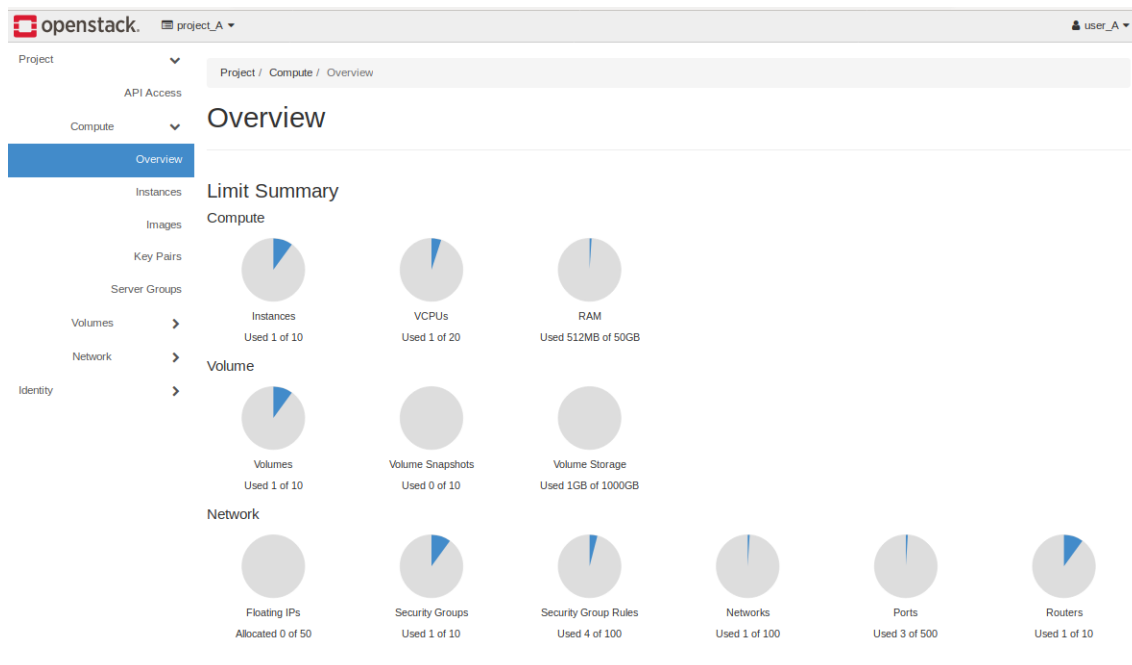


Figura 17 - Visão geral dos recursos do user_A

Feito isso, entrou-se no Openstack com o 'user_B' e a primeira característica a ser notada é que o ambiente está vazio como mostra a Figura 18, ou seja, o que foi criado pelo 'user_A' não influencia no 'user_B'. Em seguida foi feita uma configuração semelhante a do 'user_A' com uma subrede, um roteador e uma instância.

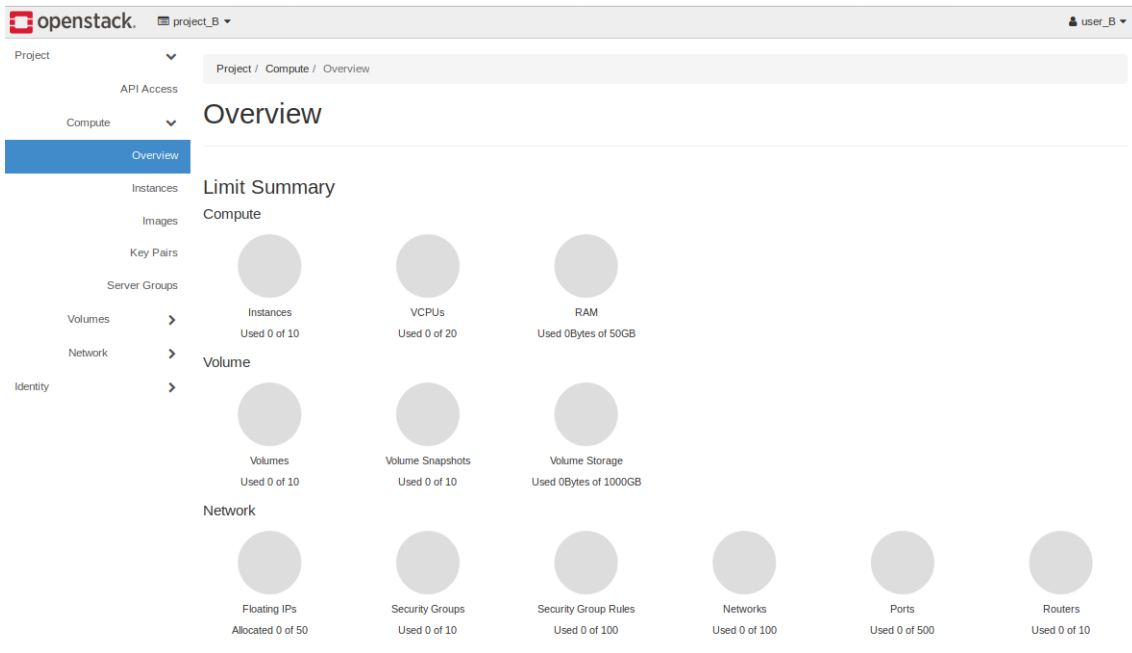


Figura 18 - Visão geral dos recursos do user_B.

Feita a configuração do user_B é possível ver a topologia da rede conforme ilustra a Figura 19.

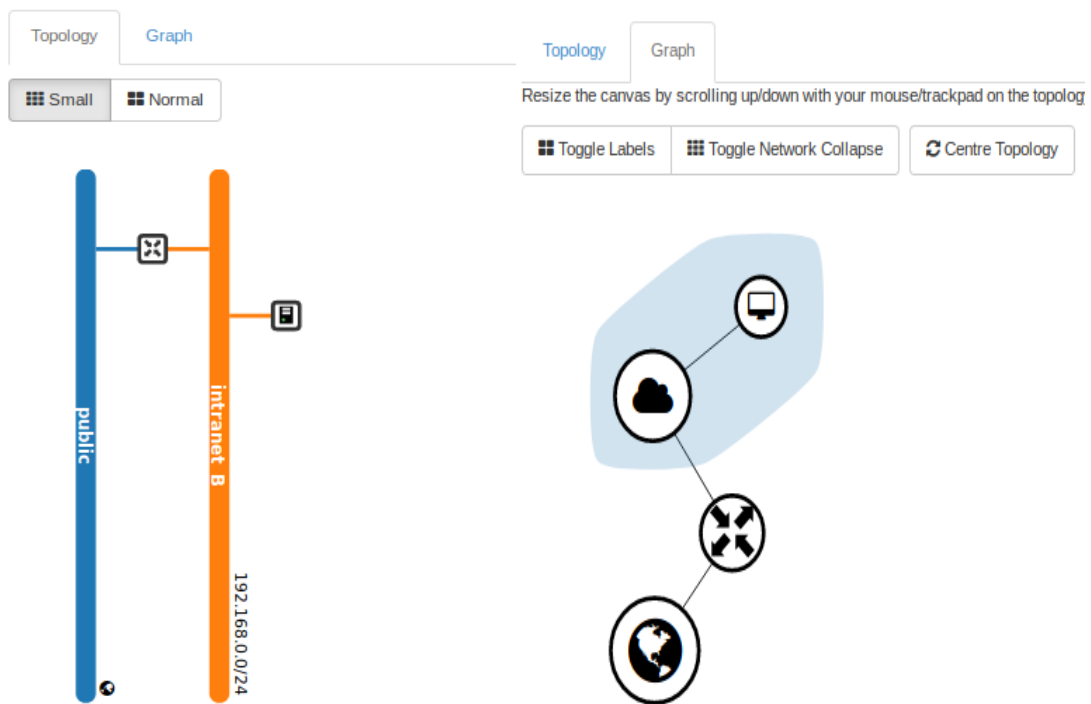


Figura 19 - Topologia do user_B.

O administrador tem acesso a todos os projetos e consegue acompanhar o uso de recursos de cada um deles, impor limites de utilização e adicionar ou remover usuários. Uma função importante é acompanhar a utilização de recursos do sistema que pode ser feita através da aba 'Admin' e posteriormente em 'Hypervisors'. Essa função monitora os três principais recursos de hardware que são vCPU, memória e disco como ilustra a Figura 20. Por meio da aba 'Instance' é possível ver todas as instâncias que estão utilizando tais recursos e a quais projetos elas pertencem como ilustra a Figura 21.

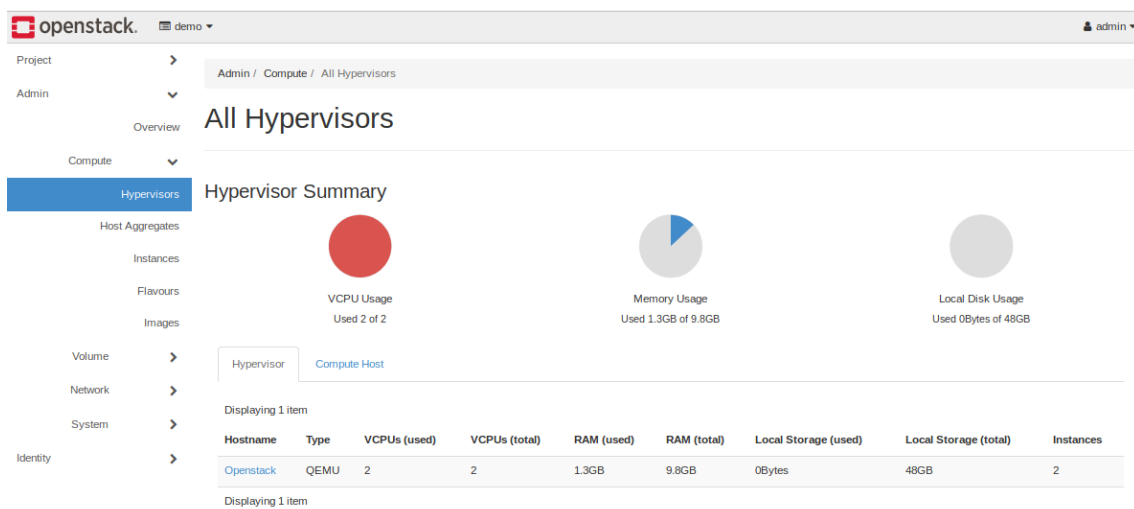


Figura 20 - Recursos utilizados geral.

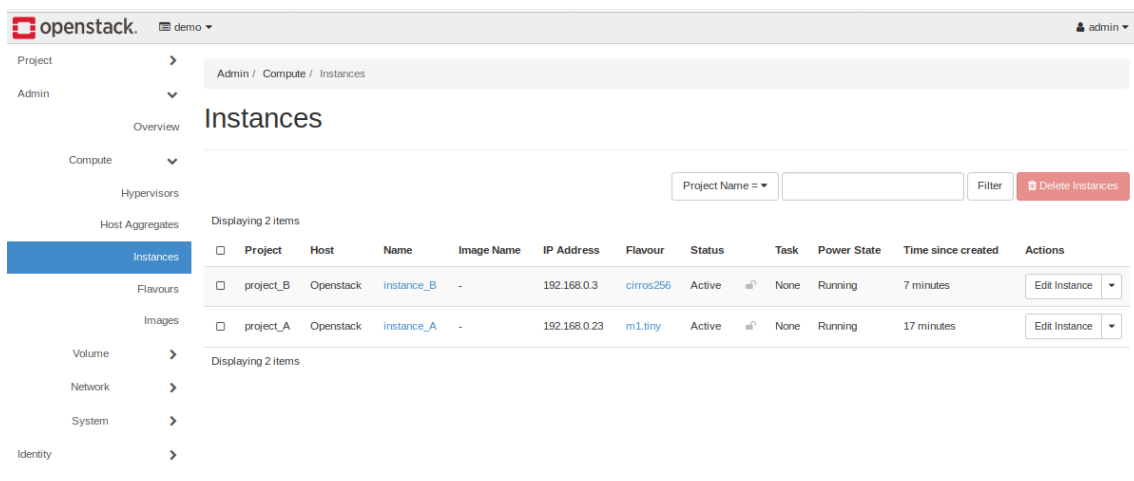


Figura 21 - Visão das instâncias pelo administrador.

5 Testes de desempenho

Um dos serviços mais utilizados na computação em nuvem é criação de instâncias, elas servem para suportar imagens de sistemas e são essenciais para a escalabilidade do ambiente. Como um dos benefícios da computação em nuvem é a agilidade em criar um novo ambiente, o tempo para uma instância estar pronta para utilização é essencial para a performance na nuvem, assim como a deletar essa mesma instância.

As instâncias podem ter diferentes configurações de hardware, também chamados de *flavors*, podendo variar o número de processadores, memória RAM e disco. Na Tabela 2 é possível observar seis tipos de instâncias com configurações diferentes de hardware diferentes. Esses *flavors* são padrões do Openstack, porém também é possível criar um *flavor* customizado de acordo com a necessidade de processador, memória RAM e disco.

Tabela 2 - Configurações de diferentes tipos de instâncias.

Tipo de instância	Configuração		
	Processador	Memória RAM	Disco
cirros256	1 vcpu	128 MB	0 GB
m1.tiny	1 vcpu	512 MB	1 GB
ds512M	1 vcpu	512 MB	5 GB
ds1G	1 vcpu	1 GB	10 GB
m1.small	1 vcpu	2 GB	20 GB
m1.medium	2 vcpu	4 GB	40 GB

O número de configurações diferentes é infinito, porém devido a capacidade do ambiente Openstack instalado foram utilizadas instâncias que se adequassem a esse cenário. Relembrando que o sistema possui ao todo 2vcpu de processamento, 10GB de memória RAM e 50GB de disco. O objetivo do teste foi avaliar o comportamento para criar e deletar diferentes *flavors* e descobrir se a diferença de hardware afeta os resultados.

5.1 Teste na visão de usuário

O primeiro teste é feito na visão de usuário, isso quer dizer que um usuário entra no portal e escolhe um *flavor* de uma instância para utilizar. Em quanto tempo a instância está liberada para poder ser utilizada?

Logo foi feito um teste de desempenho para avaliar o tempo de provisionamento para diferentes configurações de instâncias no ambiente Openstack. As instâncias utilizadas para o teste são as descritas na Tabela 2.

Cada modelo de instância foi criado 5 vezes a fim de identificar o tempo médio de cada uma para estar pronta para uso. Os testes foram feitos nas mesmas condições para todas as instâncias, ou seja, com nenhum recurso no sistema Openstack em utilização, dando condições iguais para todas as configurações. Os resultados podem ser vistos na Tabela 3.

Tabela 3 - Comparação do tempo de provisionamento de diferentes instâncias.

Tipo de instância	Tempo (s)					
	1	2	3	4	5	média
cirros256	32,0	32,9	28,4	32,5	31,2	32,0
m1.tiny	22,4	18,8	31,6	31,2	19,9	28,8
ds512M	29,0	31,4	30,2	31,5	29,9	30,2
ds1G	32,4	31,7	32,1	30,8	32,2	32,1
m1.small	32,1	32,8	31,9	31,2	31,5	31,9
m1.medium	34,7	32,3	32,8	31,5	32,1	32,3

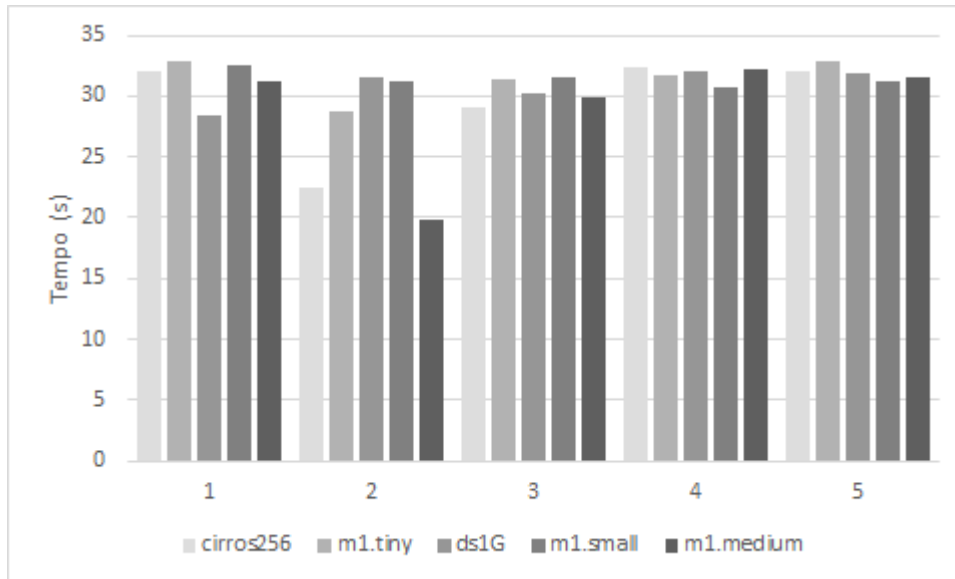


Figura 22 - Comparação do tempo de provisionamento de diferentes instâncias.

A partir dos dados obtidos da Tabela 3 e da Figura 22 foi possível notar que o tempo para provisionar as instâncias é muito equilibrado, com diferença apenas de alguns segundos e sem um padrão de distinção. Isso pode ser explicado pelo fato de que todas as instâncias precisam passar pelo mesmo processo para estar no ar, ou seja, independente da configuração da instância o tempo para ela estar pronta para uso é praticamente o mesmo.

5.2 Teste utilizando a ferramenta *Rally*

O teste de desempenho será semelhante ao feito anteriormente na visão de usuário, mas agora utilizando o Rally [17] [18]. Como o tempo para deletar uma instância é muito curto, fato que inviabilizou de ser feito na visão de usuário, mas pode ser feito com o Rally. Logo, além da comparação entre diferentes *flavors* foi possível analisar a diferença de tempo entre criar e deletar uma instância. Outra diferença para o primeiro teste na visão de usuário é que é possível criar instâncias simultaneamente conforme programado, ou seja, enquanto uma instância está terminando de ser criada ou sendo deletada uma outra está sendo criada.

Dessa forma, o teste programado foi utilizar os *flavors* da Tabela 2 com exceção do *flavor* 'm1.medium', devido ao número de processadores deste *flavor* ser 2 vcpu, ou seja a capacidade máxima do ambiente Openstack. Não seria possível criar instâncias simultaneamente, que nesse caso foram três ao mesmo tempo.

O primeiro caso foi o *flavor* 'cirros256' e os resultados podem ser vistos na Figura 23. No primeiro gráfico na parte superior é mostrado o tempo para criar e deletar a instância de cada interação, que no total foram 25. Logo, cada instância foi criada e deletada 25 vezes, dessa forma foi possível analisar melhor os tempos que foram entre 15 e 20 segundos na média total, que é um valor baixo visto que um novo ambiente pode ser criado com alguns cliques nesse curto período de tempo. O segundo gráfico inferior a esquerda mostra a distribuição de tempo para criar e deletar a instância, nesse caso por exemplo 79% do tempo foi para criar as instâncias e 21% para deletar. O terceiro gráfico mostra um histograma do número de interações pelo seu tempo de duração, nesse caso o maior número de interações durou menos de 14 segundos para serem criadas.

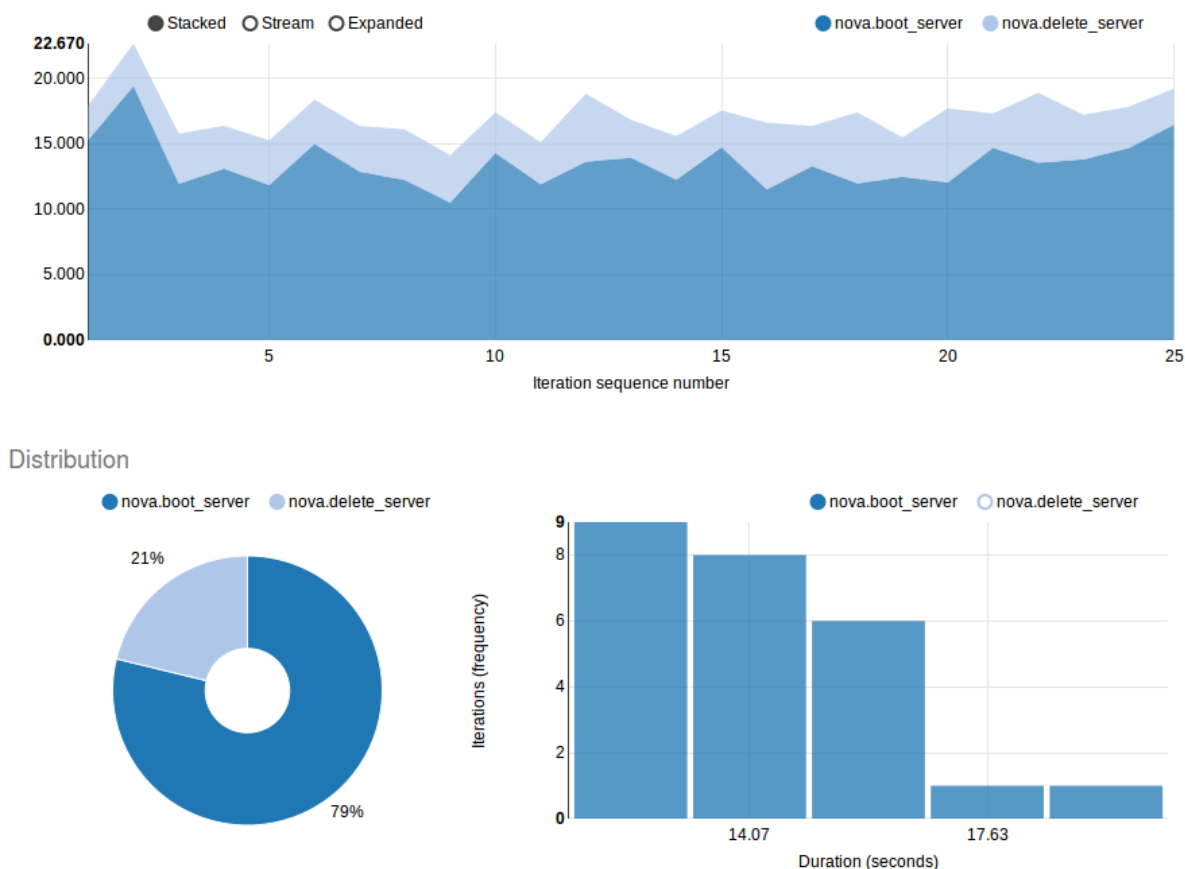


Figura 23 - Teste para criar e deletar uma instância utilizando o Rally *flavor* 'cirros256'.

O processo foi feito da mesma forma do *flavor* 'cirros256' para os *flavors* 'm1.tiny', 'ds512M', 'ds1G' e os resultados foram bem parecidos apesar das configurações diferentes porque todos eles conseguiram cumprir os requisitos de criar e deletar instâncias simultaneamente devido às suas configurações serem bastante inferiores à configuração do sistema Openstack utilizado. Isso permitiu que fossem criadas instâncias concorrentes, ou seja criadas ao mesmo tempo sem saturar a oferta de recursos de hardware. Os resultados se encontram ilustrados nas Figuras 24, 25 e 26 respectivamente.

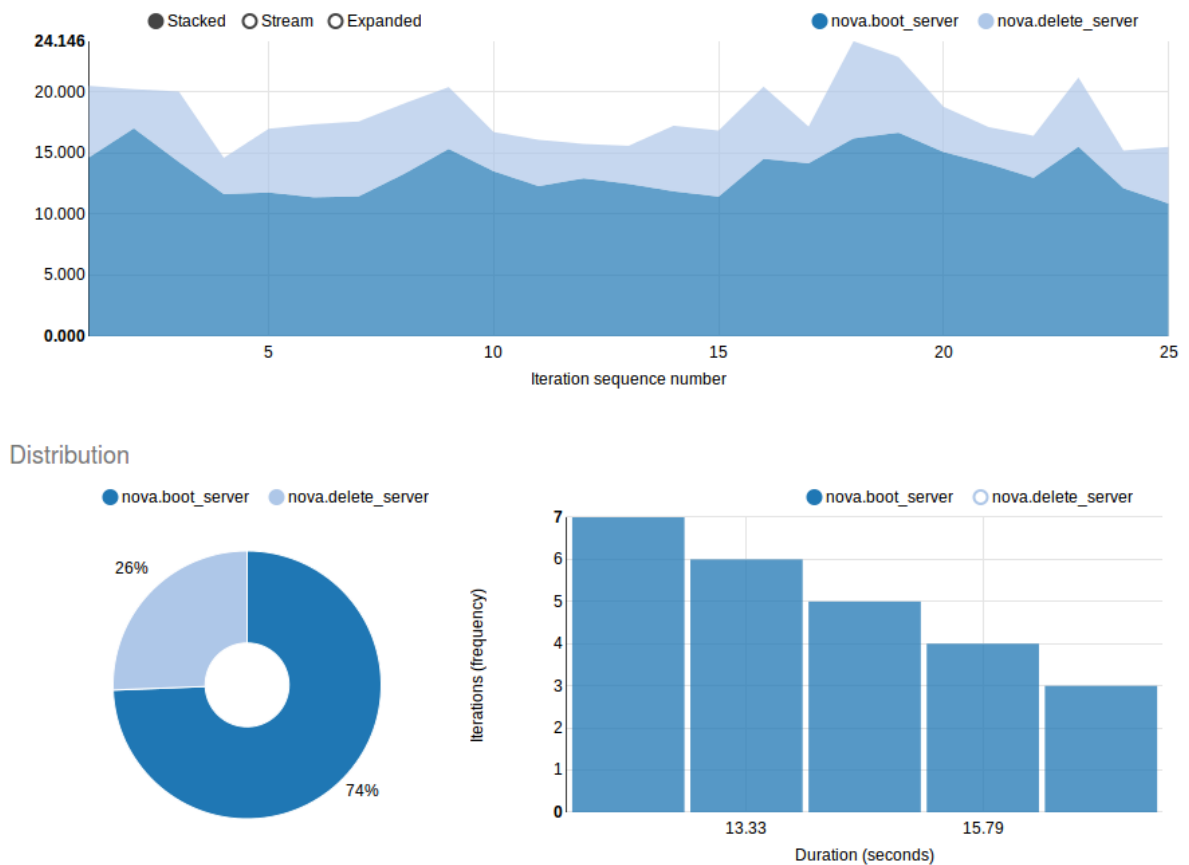
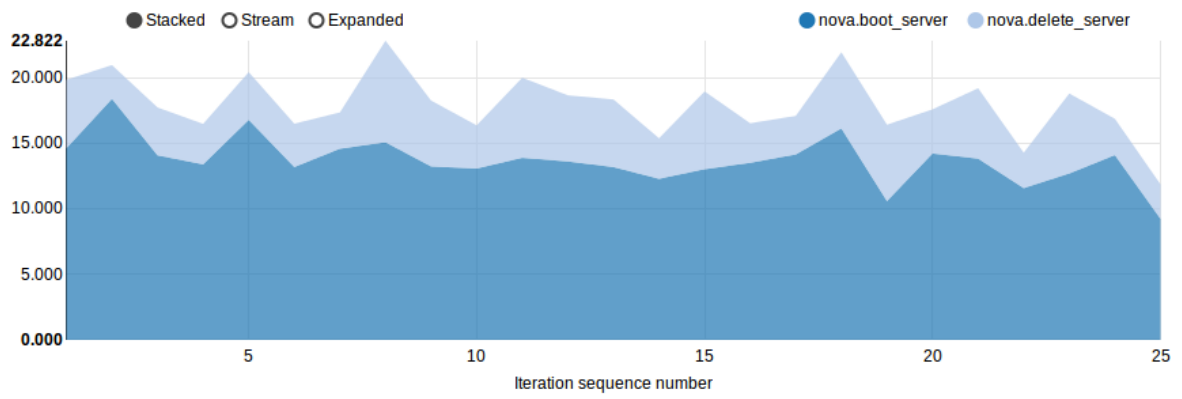


Figura 24 - Teste para criar e deletar uma instância utilizando o Rally *flavor* 'm1.tiny'.



Distribution

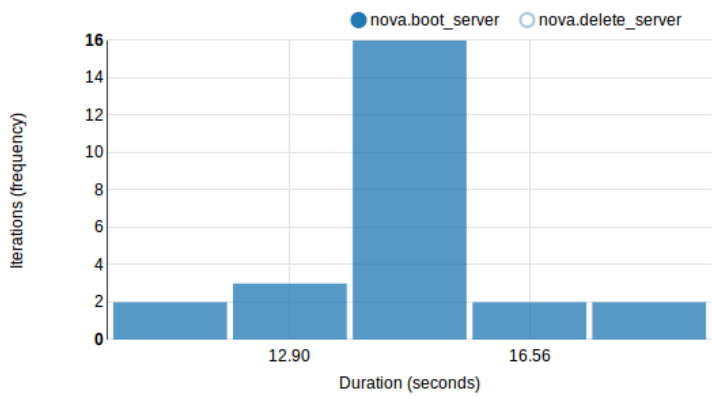
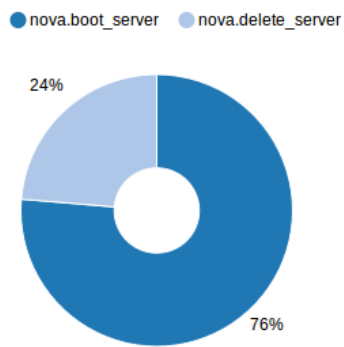
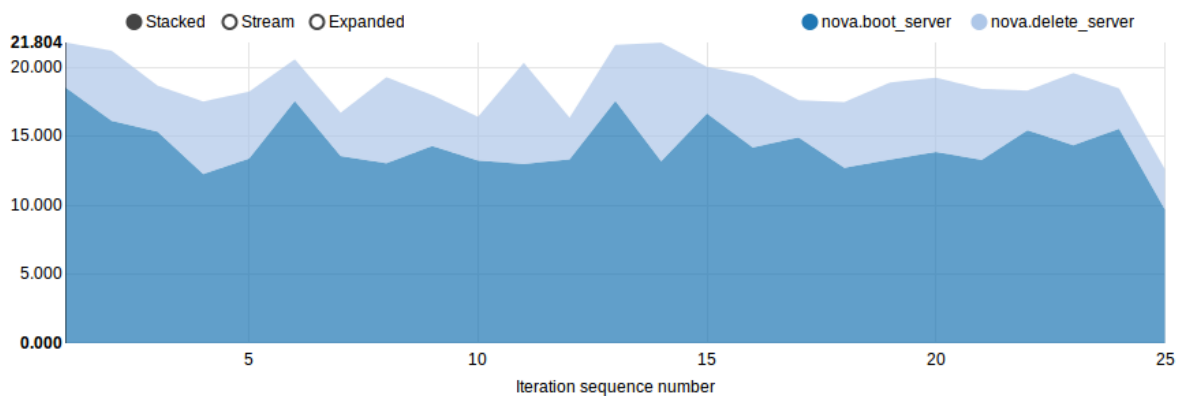


Figura 25 - Teste para criar e deletar uma instância utilizando o Rally flavor 'ds512M'.



Distribution

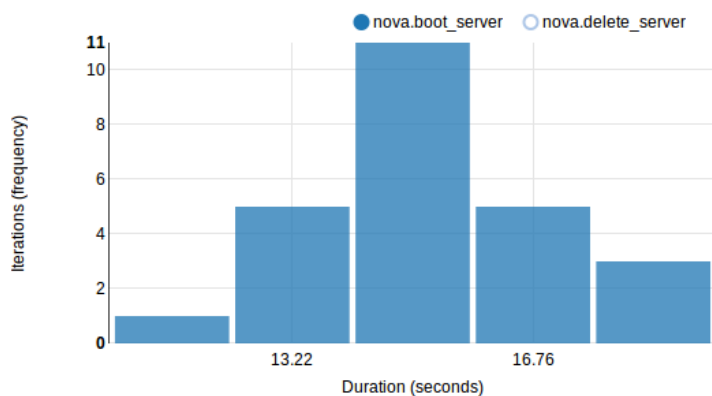
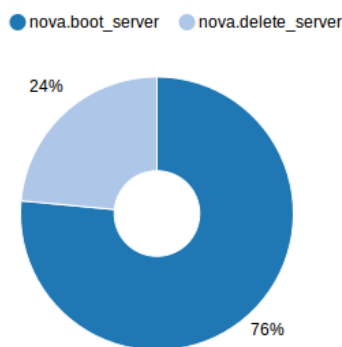
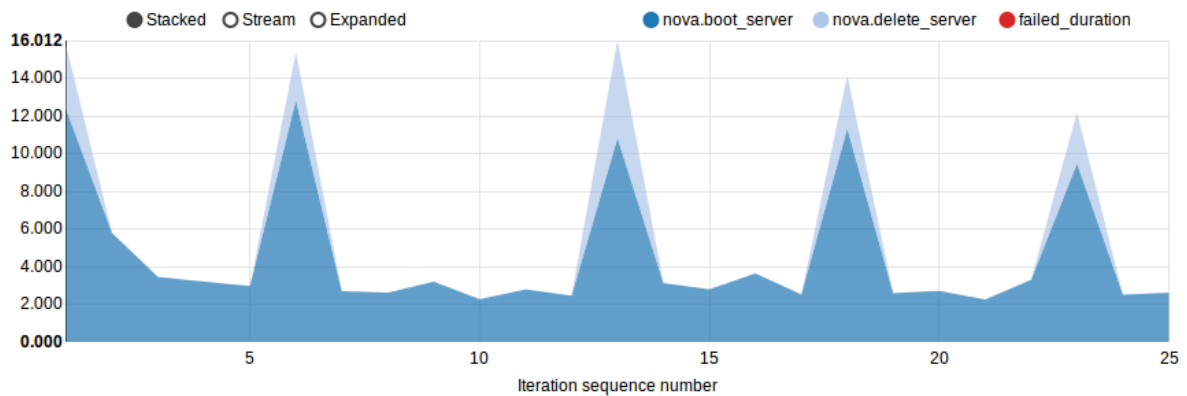


Figura 26 - Teste para criar e deletar uma instância utilizando o Rally *flavor* 'ds1G'.

O resultado do teste com o *flavor* 'm1.small' obteve um resultado diferente dos demais. Por ter 20GB de disco, não foi possível trabalhar com três instâncias concorrentes que totalizariam 60GB de disco ao mesmo tempo, uma vez que o total de disco disponível era 50GB. Nesse caso, aconteceram erros que impossibilitaram a criação de novas instâncias. Quando acontece um erro, a criação da instância é abortada e o ambiente fica disponível novamente em seguida. Por isso, em alguns momentos repetitivos não foi possível criar instâncias como pode ser visto na Figura 27. Além disso, o tempo médio foi mais baixo, uma vez que vários processos não chegaram a ser finalizados. Esse é um problema que pode acontecer em um ambiente de produção, caso a instância a ser provisionada seja maior do que a oferta do ambiente. Da mesma forma que aconteceu no teste, caso isso ocorra o processo é abortado e o recurso volta a ficar disponível para uma nova instância.



Distribution

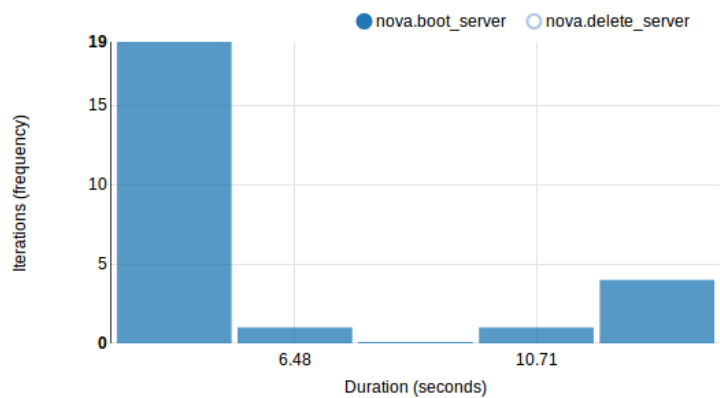
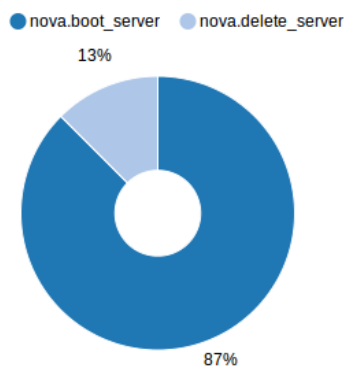


Figura 27 - Teste para criar e deletar uma instância utilizando o Rally *flavor* 'm1.small'

Na Figura 28 é possível analisar os erros que aconteceram em cada interação. No início houve um grande erro, pois, ao tentar criar instâncias simultaneamente os requisitos foram maiores que a demanda, nesses casos a instância não foi criada e recurso de hardware voltou a ficar disponível e posteriormente os erros foram ficando cíclicos em média a cada seis interações, uma vez que conforme os erros aconteciam e instâncias tinham sido deletadas os recursos eram liberados para novos testes.

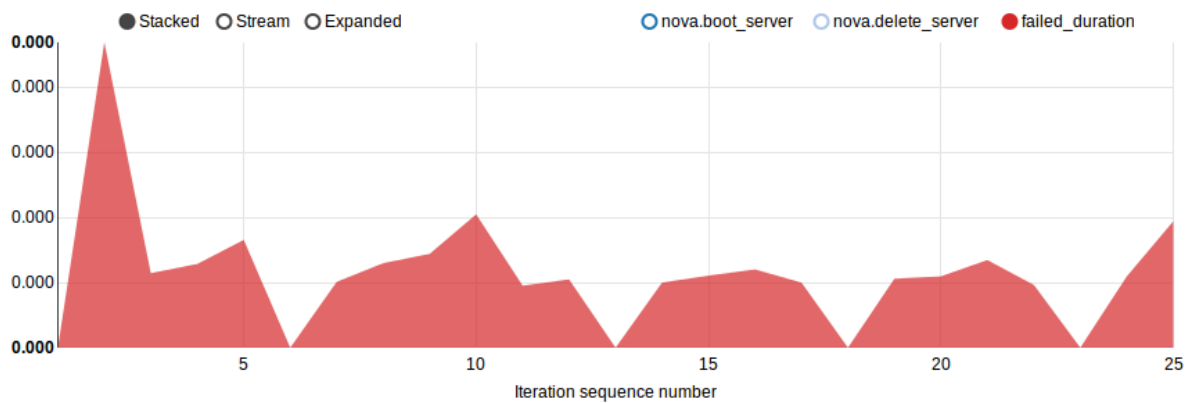


Figura 28 - Erros teste para criar e deletar uma instância utilizando o Rally *flavor* 'm1.small'.

No geral o teste de desempenho utilizando o Rally trouxe uma nova visão mais detalhada principalmente para deletar uma instância. Foi possível notar que o tempo é bem maior para criar do que para deletar uma instância uma média de 78% do tempo para criar e 22% do tempo para deletar. Um ponto interessante foi que o tempo para criar uma instância ficou abaixo do teste feito na visão de usuário, o qual foi aproximadamente 10 segundos mais rápido. Essa diferença pode estar relacionada com a disponibilidade para usuário, pois a instância estar criada para o sistema é diferente de estar disponível para o usuário utilizar, esse tempo deve ser justamente esse processo da instância criada ficar disponível para o usuário utilizar. Todavia, a média de tempo de criação das instâncias foi praticamente as mesmas para os diferentes *flavors*, resultado que também foi encontrado em [19]. Os tempos foram uma média entre 15 e 20 segundos, o que valida o primeiro teste e chegou-se a mesma conclusão que independente do *flavor* o tempo de provisionamento é praticamente o mesmo, pois a questão não são os recursos de hardware e sim o processo para ser criada.

6 Conclusão

O trabalho permitiu observar as características da computação em nuvem e entender as razões do seu grande crescimento nos modelos de computação atuais. O baixo custo e pagamento pelo uso principalmente de nuvens públicas popularizou um serviço que antigamente era apenas para poucos usuários. No entanto a nuvem pública também traz alguns desconfortos como privacidade e segurança devido ao compartilhamento de recursos, nesse aspecto a nuvem privada pode ajudar apesar do seu maior custo. Entretanto, o modelo que parece ser o ideal para boa parte dos usuários e empresas é o modelo híbrido que aproveita os benefícios das nuvens pública e privada. O Openstack é uma ferramenta ainda em desenvolvimento e aperfeiçoamento, mas já pode e é utilizada em produção e pode ajudar muito em ambientes híbridos dada a complexidade de migrar ambiente de sistemas distintos.

Os testes realizados na parte prática permitiram notar um comportamento bastante interessante da nuvem que é a agilidade. Instâncias de diferentes tamanhos levam aproximadamente o mesmo tempo para serem provisionadas, mas em todos os casos os tempos não chegam a casa dos minutos, são apenas alguns segundos e é possível ter um ambiente novo para trabalhar. A agilidade do mundo atual principalmente os projetos ágeis está diretamente associada ao modelo de nuvem, caso contrário seria inviável o desenvolvimento em tão pouco tempo.

Todavia, as restrições de hardware também restringiram o trabalho. Apesar de ser possível criar uma nuvem num simples computador pessoal os testes foram restritos principalmente a questões de processamento, memória RAM e disco. O ambiente criado foi um ambiente de desenvolvimento com baixos recursos. Num ambiente de produção outras características poderiam ser analisadas como a influência de outros usuários e instâncias no ambiente de seus vizinhos, no caso de uma nuvem pública. Se existe um ponto mais vulnerável com relação ao hardware de acordo com a aplicação, por exemplo se uma aplicação que utiliza muito processamento, memória ou disco afeta outros ambientes que compartilham do mesmo recurso. Se a latência entre ambientes instalados geograficamente separados assim como o efeito da taxa de transmissão da rede impacta na configuração e utilização do sistema.

Por fim, o trabalho foi de bastante aprendizado tanto na teoria quanto na parte prática. A computação em nuvem continua crescendo, o Openstack ainda está se desenvolvendo e a utilização de nuvens híbridas parece ser o futuro da grande maioria das empresas. Com base nos resultados alcançados e apresentados uma sugestão de continuação do trabalho é a implantação do Openstack num ambiente de produção com mais recursos de hardware, possibilitando maiores testes ou um estudo específico e aprofundado de um dos módulos do Openstack.

Referências

- [1] MELL, Peter (NIST). *The NIST Definition of Cloud Computing. Special Publication 800-145*. Setembro 2011.
- [2] Diógenes, Y., & Veras, M., *Certificação Cloud Essentials*, Rio de Janeiro: Novaterra. 2015.
- [3] *Build the future of Open Infrastructure*. [Online]. Disponível em: <<https://www.openstack.org/software/>> Acesso em: 8 Abril 2018.
- [4] KPMG India. 2011. *The Cloud: Changing the Business Ecosystem*. [Online]. Disponível em: <https://www.kpmg.com/IN/en/IssuesAndInsights/ThoughtLeadership/ThT_CloCl_Changing_the_Business_Ecosystem.pdf> Acesso em: 10 Maio 2018.
- [5] Veras, M. *Computação em nuvem*. Rio de Janeiro: Brasport. 2015
- [6] BUI, Tuan-Anh. *Cloud Network Performance Analysis: An OpenStack Case Study*. Janeiro 2016.
- [7] MICROSOFT. *The economics of the Cloud*. Microsoft. 2010.
- [8] CHAPPELL, David. *The benefits and risks of Cloud Platforms: a guide for business leaders*. David Chappell & Associates. 2011
- [9] IBM. *A Practical Approach to Cloud IaaS with IBM SoftLayer: Presentations Guide Deploying Openstack*. RedBooks. 2016
- [10] Software - OpenStack is open source software for creating private and public clouds. [Online]. Disponível em: <<https://www.openstack.org/software/>> Acesso em: 8 Abril 2018.
- [11] Introdução ao OpenStack. [Online]. Disponível em: <<https://www.redhat.com/pt-br/topics/openstack>> Acesso em: 8 Abril 2018.
- [12] OpenStack Docs: DevStack. [Online]. Disponível em: <<https://docs.openstack.org/devstack/latest/>> Acesso em: 30 Abril 2018.

- [13] Overview — rally 0.9.1.dev386 documentation. [Online]. Disponível em: <<https://docs.openstack.org/developer/rally/overview/overview.html>> Acesso em: 24 Outubro 2018.
- [14] Overview — rally 0.9.1.dev386 documentation. [Online]. Disponível em: <<https://docs.openstack.org/developer/rally/overview/overview.html#use-cases>> Acesso em: 24 Outubro 2018.
- [15] OpenStack on Ubuntu is your scalable private cloud, by Canonical | Ubuntu. [Online]. Disponível em: <<https://www.ubuntu.com/openstack>> Acesso em: 7 Abril 2018.
- [16] OpenStack Docs: All-In-One Single Machine. [Online]. Disponível em: <<https://docs.openstack.org/devstack/latest/guides/single-machine.html>> Acesso em: 30 Abril 2018.
- [17] What is Rally? — rally 0.9.1.dev386 documentation. [Online]. Disponível em: <<https://docs.openstack.org/developer/rally/>> Acesso em: 24 Outubro 2018.
- [18] GitHub - openstack/rally-openstack: A collection of plugins for Rally framework designed for the OpenStack platform. [Online]. Disponível em: <<https://github.com/openstack/rally-openstack/>> Acesso em: 24 Outubro 2018.
- [19] PANZNER, T., TORNYAI, R., BALAZS, G., SCHMIDT, A., KERTESZ, A., *Performance Analysis of an OpenStack Private Cloud*. Janeiro 2016.

Anexo

Nesta seção de anexo serão apresentados passo a passo os *scripts* utilizados para instalação do Openstack utilizando o Devstack e também os testes de desempenho utilizando o Rally.

O ambiente Openstack é configurado e manipulado através da rede, logo é preciso que o ambiente de instalação tenha um IP fixo e um range reservados de IP's, os quais serão provisionados todos os serviços.

Com os IP's previamente alocados. O primeiro procedimento é criar um usuário, que nesse caso foi chamado de stack:

```
>sudo adduser stack
```

É preciso entrar no modo *root* para habilitar o usuário 'stack' como um usuário privilegiado acessando o sudo a acrescentando as informações abaixo de "*User privilege specification*":

```
>su -  
>visudo  
(...)  
stack ALL=(ALL) NOPASSWD: ALL
```

O próximo passo é acessar o sistema com o usuário 'stack' e fazer algumas atualizações e instalar o *git* que será utilizado para fazer clone das pastas do Devstack e do Rally:

```
>sudo apt-get update -y  
>sudo apt-get install sudo -y  
>sudo apt install git
```

Posteriormente é feito o clone das pastas do Devstack e do Rally:

```
>git clone https://git.openstack.org/openstack-dev/devstack
>git clone https://github.com/openstack/rally
```

O passo seguinte é acessar a pasta 'devstack' e criar o arquivo 'local.conf' que guiará a instalação do Openstack:

```
>cd devstack
>vim local.conf
```

Esse arquivo é fundamental para o sucesso da instalação e de todas as suas funcionalidades. Primeiramente é habilitado o serviço do Rally para já iniciar com autenticação do Openstack. Em seguida são adicionados o IP do *host*, que é o IP fixo conforme comentado no início da seção e os IP's flutuantes e fixos para serem alocados na plataforma. É preciso adicionar o tamanho e principalmente a interface de rede que está sendo utilizada. Por fim, basta colocar algumas credenciais do administrador, banco de dados e serviços como mostra o *script* abaixo:

```
[[local|localrc]]
enable_plugin rally https://github.com/openstack/rally master
HOST_IP=10.0.2.15
FLOATING_RANGE=192.168.1.0/24
FIXED_RANGE=10.1.1.0/24
FIXED_NETWORK_SIZE=256
FLAT_INTERFACE=enp0s3
ADMIN_PASSWORD=supersecret
DATABASE_PASSWORD=iheartdatabases
RABBIT_PASSWORD=flopsymopsy
SERVICE_PASSWORD=iheartksl
```


Depois de salvar o arquivo 'local.conf' com as instruções acima o processo de instalação pode iniciar com o comando abaixo:

```
>./stack.sh
```

O Devstack automatiza toda a instalação, é preciso apenas acompanhar caso apareça algum erro no meio do processo. Com as configurações de hardware utilizadas o tempo de instalação foi de aproximadamente 50 minutos.

Terminada a instalação é possível acessar a plataforma através do navegador pelo endereço <'HOST_IP'/dashboard>. Nesse caso o 'HOST_IP' é o IP fixo configurado no arquivo 'local.conf'.

Para realizar os testes do Rally é preciso criar um arquivo '.json' com as especificações do teste. Abaixo segue o exemplo de um *script* de um teste utilizado no trabalho onde são criadas e deletadas instâncias para medir o tempo desse processo:

```
{% set flavor_name = flavor_name or "m1.tiny" %}
{
  "NovaServers.boot_and_delete_server": [
    {
      "args": {
        "flavor": {
          "name": "{{flavor_name}}"
        },
        "image": {
          "name": "^cirros.*-disk$"
        },
        "force_delete": false
      },
      "runner": {
        "type": "constant",
        "times": 25,
        "concurrency": 2
      },
      "context": {
        "users": {
          "tenants": 3,
          "users per tenant": 2
        }
      }
    }
  ]
}
```

```

    }
    },
    "sla": {
      "failure_rate": {
        "max": 0
      }
    }
  },
  {
    "args": {
      "flavor": {
        "name": "{{flavor_name}}"
      },
      "image": {
        "name": "^cirros.*-disk$"
      },
      "auto_assign_nic": true
    },
    "runner": {
      "type": "constant",
      "times": 25,
      "concurrency": 2
    },
    "context": {
      "users": {
        "tenants": 3,
        "users_per_tenant": 2
      },
      "network": {
        "start_cidr": "10.2.0.0/24",
        "networks_per_tenant": 2
      }
    },
    "sla": {
      "failure_rate": {
        "max": 0
      }
    }
  }
]
}

```

Após criar o arquivo basta iniciá-lo com o comando abaixo para o teste de desempenho começar:

```
>rally task start 'arquivo_teste_desempenho'.json
```

Finalizado o teste o Rally oferece um dashboard para acompanhar e analisar o teste executado, para gerar o *report* e visualizá-lo no navegador basta inserir o *script* abaixo:

```
>rally task report `protocolo_report` --out 'nome_do_teste'.html  
>rally task report --out='nome_do_teste'.html --open
```

Esses são os principais passos para configurar um ambiente Openstack através do Devstack e criar um teste de desempenho utilizando o Rally.