UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PROPOSTA DE VIM *ON-DEMAND* PARA FATIAMENTO DE NUVEM

EDUARDO HENRIQUE ROCHA ZANELA

ORIENTADOR: PROF. DR. FÁBIO LUCIANO VERDI

São Carlos – SP

Fevereiro/2019

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PROPOSTA DE VIM *ON-DEMAND* PARA FATIAMENTO DE NUVEM

EDUARDO HENRIQUE ROCHA ZANELA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Sistemas Distribuídos e Redes de Computadores.
Orientador: Prof. Dr. Fábio Luciano Verdi

São Carlos – SP

Fevereiro/2019

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Eduardo Henrique Rocha Zanela, realizada em 20/02/2019:

Prof. Dr. Fabio Luciano Verdi
UFSCar

Prof. Dr. Paulo Matias
UFSCar

Prof. Dr. Cesar Augusto Cavalheiro Marcondes
ITA

Dedico o resultado desse trabalho a todos que me apoiaram durante a minha caminhada no período do mestrado, minha família, em especial aos meus sobrinhos Davi, Heitor e Otávio, a todos do laboratório Asgard e a todos apaixonados por computação.

AGRADECIMENTOS

Agradeço primeiramente à Deus pelo dom da vida e por sempre me dar forças durante os momentos difíceis, fazendo jamais desistir. Aos meus pais por sempre me apoiarem e por tudo que fizeram por mim. Ao Prof. Dr. César Marcondes por ter confiado e dado todo apoio durante o período do mestrado. A todos que fazem parte do Departamento de Computação da Universidade Federal de São Carlos e, principalmente, aos que frequentam e fazem parte do laboratório Asgard, pois este local se tornou a minha segunda casa. Em especial meus amigos Emerson Barea, William Akihiro, Lucian Beraldo e Alex Marino. Agradeço a todos os amigos que fiz durante o período que estive na cidade de São Carlos e, por fim, não menos importante, à República Vegas que me proporcionou conhecer pessoas que se tornaram mais do que amigos, mas sim uma família, em especial meus amigos João Castro e Diego Pedroso por diversas histórias juntos.

Look
If you had
One shot
Or one opportunity
To seize everything you ever wanted
In one moment
Would you capture it
Or just let it slip?
Lose Yourself - Eminem

RESUMO

A inserção das redes 5G no âmbito das indústrias exigem um apresto para que suas infraestruturas suportem eficientemente numerosos e variados serviços, uma das alternativas para isto é o fatiamento da rede. O fatiamento da rede permite às operadoras de redes fornecerem, simultaneamente, diversas redes virtuais dedicadas com funcionalidades específicas, sobre uma infraestrutura de rede comum, possibilitando uma variedade de serviços para diferentes casos de usos. Este trabalho apresenta uma proposta de VIM Sob-Demanda para fatiamento de nuvens que permita controle Bare Metal e de serviços de redes, com enfoque na sua criação sob demanda. Para isto, foi implementado um sistema de gerenciamento utilizando o Openstack em conjunto com ferramentas de virtualização, permitindo a instanciação de uma máquina física. Nesta pesquisa, apresentamos 3 opções de instalação: Instanciação Completa Bare Metal, Instanciação Completa com Containers e Instanciação Completa com Imagem Pré-Carregada. Os resultados indicam que a arquitetura proposta, implementada, orquestrada e testada atende os requisitos do VIM On-Demand, e apontam os principais pontos preponderantes no VIM.

Palavras-chave: Computação em nuvem, Fatiamento de rede, Função de rede, Infraestrutura, Fatia como Serviço.

ABSTRACT

The evolution of mobile communication is opening up opportunities for the innovation of vertical industries, whose varied applications demand different requirements. The insertion of 5G networks in vertical industries requires an infrastructure preparation to efficiently support numerous and varied services, one of the alternatives for this is the network slicing. Network slicing allows network operators to simultaneously provide a number of dedicated virtual networks with specific functionalities over a common network infrastructure, allowing a variety of services for different applications. This paper presents a proposal for a Virtualized Infrastructure Manager On-Demand for cloud slicing that allows bare metal control and network services with focus in on-demand creation. For this, a management system was implemented using Openstack in conjunction with virtualization tools allowing the instantiation of a physical machine. In this paper, we present 3 options of installation: complete instantiation bare metal, complete instantiation with containers and complete instantiation with preloaded image. The results indicate that the proposed, implemented, orchestrated and tested architecture meets the requirements of VIM On-Demand, and points out the main prevailing points in VIM.

Keywords: Cloud computing, Network slicing, Network function, Infrastructure, Slice as a Service.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Capítulo 1

INTRODUÇÃO

Um dos maiores desafios enfrentados no desenvolvimento da rede 5G é atender às diversas demandas de consumidores por novos serviços, modificando a forma de implantação de redes de tamanhos fixos para tamanhos variáveis, possibilitando atender aos requisitos de desempenho específicos quanto à latência, escalabilidade, disponibilidade e confiabilidade, impostos por cada caso de uso. Com o objetivo de colaborar no desenvolvimento da tecnologia 5G, o projeto *Novel Enablers for Cloud Slicing* (NECOS) propõe o desenvolvimento de uma plataforma que permite o oferecimento do fatiamento de rede como um serviço (*Slice as a Service*).

Este Capítulo apresenta uma introdução à proposta do trabalho e está dividido em seções com a seguinte estrutura: Seção 1.1 - contextualiza e relaciona o tema principal da pesquisa na atualidade; Seção 1.2 - identifica e apresenta o problema tratado; Seção 1.3 - traz a motivação para o desenvolvimento do presente trabalho e aponta os principais objetivos da pesquisa; Seção 1.4 - apresenta a estrutura do restante do trabalho.

1.1 Contexto

Com a evolução das tecnologias das comunicações móveis, aumentou-se a necessidade de melhorar o desempenho dos serviços oferecidos, pois, a estrutura que anteriormente atendia principalmente a comunicação humana, agora atende outros formatos, como as comunicações de voz e dados industriais, contribuindo para a transformação digital global.

Diante disto o 5G tem como plano futuro fazer parte do âmbito das indústrias, tendo como foco principal as verticais, satisfazendo as demandas de consumidores por novos serviços e negócios, modificando a forma de implantação de redes de tamanhos fixos para tamanhos variáveis, possibilitando atender aos requisitos de desempenho específicos quanto à latência,

escalabilidade, disponibilidade e confiabilidade, impostos por cada caso de uso. Contudo, a mudança de paradigma de oferecimento de uma estrutura de rede de tamanho fixo para variável não é algo trivial.

Nessa linha, Han et al. (2015) explica que a substituição de equipamentos físicos por outros do mesmo tipo com algumas funções a mais, ou apenas com maior capacidade para melhorar o desempenho pelas operadoras telefônicas, mesmo que ainda seja suficiente para a maioria das finalidades, ainda ocasiona dificuldade em escalonar os serviços oferecidos.

A necessidade de dar suporte as indústrias que detém casos de usos diversificados, necessitando de mais requisitos e recursos específicos é um dos principais estimuladores no desenvolvimento dos sistemas 5G, e para atender a crescente demanda de serviços sobre a mesma infraestrutura de rede, é consenso que os sistemas 5G estabeleça aprimoramentos em sua arquitetura com relação às implementações atuais.

Ainda segundo Han et al. (2015), a inserção de novos serviços nas redes atuais sem realizar alguma mudança nesses sistemas podem ocasionar problemas como aumento de custo, necessidade de maior espaço físico para a infraestrutura, alto consumo de energia e a imposição de profissionais qualificados.

Uma das alternativas que vem sendo utilizada em outras situações e pode ser adotado no avanço do 5G é a virtualização da rede. A virtualização possibilita transformar as redes reais em redes virtuais utilizando soluções tecnológicas baseadas principalmente em software, como por exemplo as Redes Definidas por Software (*Software Defined Networking* - SDN), Virtualização das Funções de Rede (*Network Functions Virtualization* - NFV) e as Funções de Redes Virtualizadas (*Virtualized Network Function* - VNF), com isso permitindo maior capacidade de programação, flexibilidade e modularidade para o desenvolvimento de diferentes redes virtuais para diferentes casos de uso.

Segundo Taurion (2009) a computação em nuvem permite aumentar e flexibilizar os recursos computacionais, possibilitando utilizar infraestruturas tecnológicas complexas, onde os usuários não necessitam realizar a instalação, configuração e atualização de softwares para terem acesso aos serviços solicitados. Ainda nessa linha, Clayman (2017) explica que em cada rede virtual de uma indústria, o locatário ou cliente pode ser considerado como uma fatia de rede, e sua definição pode ser dado como redes lógicas executadas em rede física ou virtual, mutuamente isoladas, com controle e gerenciamento independentes, que podem ser criados sob demanda.

Com objetivo de reduzir as limitações das tecnologias e infraestrutura das redes atuais, al-

gumas instituições brasileiras e europeias reuniram-se em um projeto denominado de NECOS, visando desenvolver uma plataforma que ofereça *slices* gerenciáveis de recursos virtualizados de rede, computação e armazenamento, para diferentes cenários, permitindo a execução de funções que originalmente eram executadas na infraestrutura dos clientes, passem a ser executadas nas infraestruturas computacionais das operadoras.

Para tal, é necessária a introdução do fatiamento de rede baseado em demandas, que permitirá que cada fatia de rede, denominado como *slice*, seja alocada conforme a demanda e requisitos do locatário em função de suas aplicações. O ponto central da implantação do fatiamento é dependente da orquestração de atividades que possibilitem a instalação, preparação do ambiente sucedida de suas configurações e instalação dos serviços requeridos, permitindo enfim desfragmentar uma infraestrutura total em divergentes infraestrutura sob demanda.

A conjuminação destas tarefas visa, por fim, a elaboração de um aparato de gerenciamento de fatias de redes virtualizadas, as quais adotam anglicismo *Virtualized Infrastructure Manager* (VIM).

O presente trabalho traz como proposta o desenvolvimento de um VIM sob demanda (VIM *on-demand*), apresentando seu desenvolvimento de maneira virtualizada e 3 opções de instalações: Instanciação Completa *Bare Metal*, Instanciação Completa com *Containers* e Instanciação Completa com Imagem Pré-Carregada.

1.2 Problemática

Dentre os aspectos importantes correlacionados ao desenvolvimento do VIM, são as verificações dos pontos preponderantes relativos à escalabilidade, aferições pertinentes à resiliência do ambiente e constatações dos tempos gastos durante o processamento de cada etapa da instalação. A proposta apresentada neste trabalho aponta para o desafio de um VIM de fatiamento de rede que criem *slices* sob demanda. Espera-se que os *slices* caracterizem leveza, praticidade, facilidade de instalação e configuração, somando-se ainda à capacidade de gerenciamento dos componentes de infraestruturas virtualizadas. Ao término deste trabalho, espera-se responder duas questões: (*i*) Os recursos computacionais, tais como memória principal e quantidade de núcleos são determinantes para a eficiência do processo? (*ii*) É possível aferir vantagem em desempenho da abordagem VIM *on-demand* em relação à tradicional?

1.3 Motivação e Objetivos

Os serviços de computação em nuvem quando comparada às redes físicas, oferece benefícios relevantes (gerenciamento dos recursos, redução de custos e manutenção) que refletem em uma maior qualidade e eficiência aos serviços ofertados, porém, ainda assim, existem limitações que devem ser exploradas e melhoradas, como a restrição de acesso *bare metal* ou a limitação de softwares específicos para o controle da nuvem.

Com propósito de sanar essas deficiências vários estudos e pesquisas propõem o desenvolvimento e padronização das redes 5G, dentre esses estudos, o NECOS tem como base incluir a capacidade de responder à demanda de novos serviços com um novo modelo de gerenciamento das redes através do *Slice as a Service*. O *Slice as a Service* é responsável por dividir a infraestrutura necessária conforme o pedido do locatário em apenas um *slice*, aglomerando os recursos gerenciados como um todo, permitindo acomodar componentes de serviço independentemente de outros *slices*.

Dividir uma infraestrutura em diversos *slices* permite eliminar as restrições e limitações de serviços específicos com relação à infraestrutura do provedor que o oferece, proporcionando executar diferentes casos de uso com diversos propósitos em uma única infraestrutura de um provedor, além de possibilitar o oferecimento de recursos que podem ser ofertados por multi provedores.

Para isso, é necessário que o orquestrador do *slice* atribua um VIM que possa realizar a gestão completa dos recursos de maneira compartilhada, um software que viabilize no desenvolvimento de uma infraestrutura como serviço, permitindo ser implementado em *Data Center (DC) slice Controller*, ao invés de ser implementado por rede, foco contrário do projeto NECOS diante da motivação do 5G. Implementar o Openstack no *DC* faz com que os recursos de uma infraestrutura em cloud computing já existente possam ser compartilhados com as demais aplicações. Um dos fortes elementos do Openstack é a sua arquitetura, que se encontra subdividida em diversos módulos correspondentes aos principais serviços e componentes, cada qual para a sua tarefa, permitindo assim maior e melhor flexibilidade, escalabilidade e versatilidade.

No melhor dos nossos esforços, este trabalho apresenta uma proposta de desenvolvimento de um VIM sob demanda (VIM *on-Demand*), no qual será apresentado o desenvolvimento de forma virtualizada e dividido em 3 opções de instalações: Instanciação Completa *Bare Metal*; Instanciação Completa com *Containers* e Instanciação Completa com Imagem Pré-Carregada. Atribui-se como aspecto inovador desta proposta (*i*) o uso de equipamentos de baixo custo, (*ii*) uso exclusivo de tecnologias de código aberto, (*iii*) orquestração total da implementação e (*iv*)

instanciação de diferentes casos de uso. Ressalta-se a escolha da utilização do Openstack em função dos seus serviços aderentes ao propósito deste trabalho, que são a existência do *bare metal* e serviços de controle de computação.

1.4 Organização do Trabalho

Este trabalho está dividido em 5 capítulos:

No Capítulo 2 é apresenta a fundamentação teórica que baseia o trabalho. Nele são abordados os conceitos de fatiamento de rede, virtualização das funções de rede, redes definidas por software, finalizando com a apresentação do projeto NECOS.

No Capítulo 3 é apresentada uma visão geral do gerenciador de infraestrutura virtualizado, com três propostas de instalação.

No Capítulo 4 são apresentados os trabalhos relacionados que apontam o estado da arte atual referente ao tema principal dessa pesquisa.

No Capítulo 5 são apresentadas tecnologias, ambientes e os métodos utilizados durante a implementação do trabalho, por fim, são apresentados os experimentos e resultados, comparando o tempo de resposta e desempenho.

O Capítulo 6 traz a conclusão do trabalho e sugestões para trabalhos futuros.

Capítulo 2

FUNDAMENTAÇÃO TEÓRICA

Este Capítulo retrata os materiais, métodos, conceitos e definições utilizados durante o desenvolvimento da proposta; na Seção 2.1 são apresentados os principais conceitos e definições envolvidos neste trabalho, estando eles apresentados na seguinte sequência: Subseção 2.1.1 - apresenta os conceitos de virtualização e suas tecnologias empregadas no fatiamento de rede para oferecimento do *Slice as a Service*; Subseção 2.2 - apresenta os conceitos relacionados ao fatiamento de rede física; Subseção 2.2.1 - apresenta os conceitos do *Slice as a Service* e apresenta um exemplo de caso de uso e finalizando com a Seção 2.3 - apresentando brevemente o projeto NECOS.

2.1 Conceitos e Definições

Para efetivar o desenvolvimento da proposta de *VIM on-demand* nas condições propostas, alguns requisitos devem ser previamente elencados e discutidos. Os requisitos que nortearam o processo de implementação, configuração e instanciação do ambiente e experimentos, seguem descritos:

2.1.1 Virtualização

A virtualização permite criar uma versão similar de algo sendo estes recursos físicos ou já virtualizados, desde que suporte um padrão recursivo com diferentes camadas de abstração, como por exemplo, um software adaptável como sistemas operacionais, dispositivos, hardware ou recursos de redes. A computação em nuvem, faz intensa utilização do conceito da virtualização, uma vez que a mesma proporciona uma redução de custo e melhor controle de seu ambiente, segregando a aplicação e sistema operacional dos componentes físicos de origem.

Virtualizar os serviços de rede implica na transformação de suas funções de redes estáticas em funções virtuais. Para (HAN et al., 2015), a virtualização das funções de redes propõem melhorar o provisionamento dos serviços de redes. A transformação de redes reais em redes virtuais pressupõe a aplicação de tecnologias baseadas em software, conferindo capacidade de controle, flexibilidade e programação, além de colaborar com o desenvolvimento de diferentes redes virtuais para diferentes casos de uso.

Para o fatiamento de rede, é fundamental empregar a virtualização, pois a mesma permite criar múltiplas redes virtuais separadamente da rede física, compartilhar os mesmos recursos físicos caso necessário para a mesma *slice*, além de simplificar e melhorar o gerenciamento de recursos (ORDONEZ-LUCENA et al., 2017).

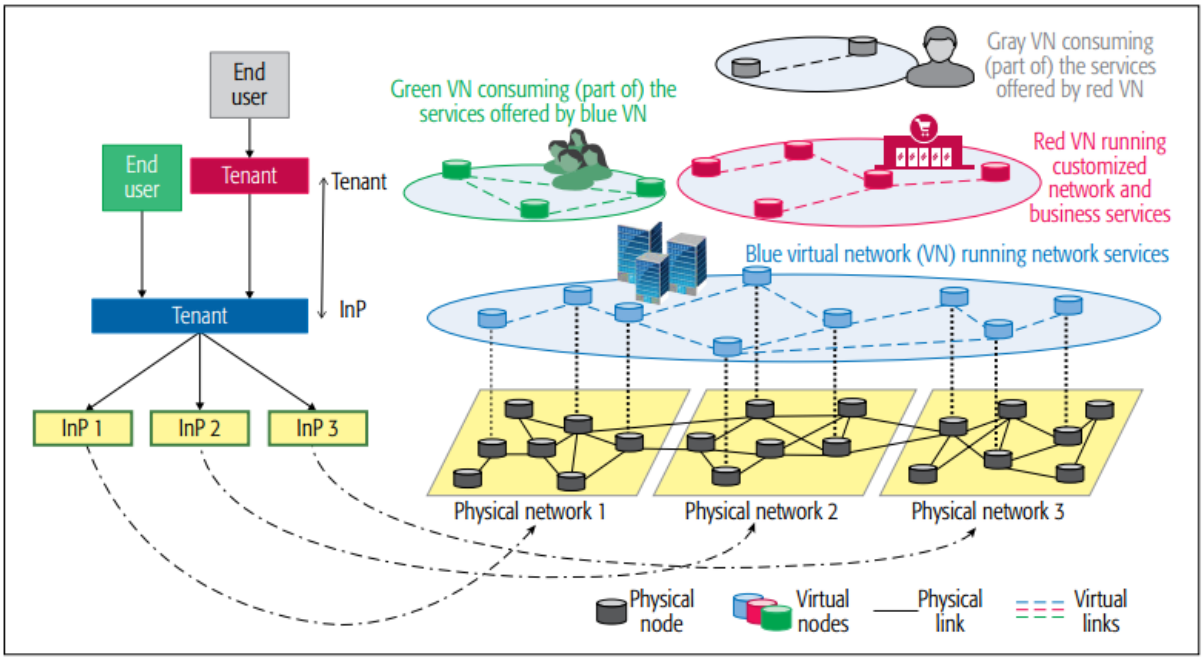


Figura 2.1: Atores e funções da rede de fatiamento 5G.

Fonte: (ORDONEZ-LUCENA et al., 2017).

Utilizar a recursividade diretamente na virtualização da estrutura permite que os dois primeiros agentes (locatário e provedor) possam atuar num padrão de multicamadas, sendo um locatário em um caso de uso e um provedor de infraestrutura na camada imediatamente acima, ou a oferta de serviços entre inquilinos (*Tenants*) (Figura 2.1), onde nesse caso o segundo inquilino forneceria serviços de rede mais avançados para seus próprios usuários.

Ainda analisando a Figura 2.1, é possível verificar que cada inquilino recebe uma ”mini-nuvem”de serviços, visto o que até o presente momento era permitido apenas acesso e controle dos serviços superiores, e no presente momento é capaz e permitido acessar, verificar e controlar

a nível *bare metal*, devido cada nuvem estás divididas em *slices*, que por sua vez são totalmente isolados aos demais. Cada *slice* e seu VIM associado são independentes das outras fatias e outros VIMs.

Em relação ao compartilhamento de rede, não existe a necessidade de utilizar uma única rede física pelo provedor, pois assim é permissível compartilhamento de recursos e entregar a melhor rede física para atendimento do locatário, eliminando problemas na rede como gargalos, perda de pacotes e lentidão.

Para a transformação das funções de redes estáticas em funções virtuais, é necessário tecnologias virtuais que permitam desacoplar as funções de rede dos dispositivos de hardware dedicados e as transfiram para serem alojadas em máquinas virtuais (*Virtual Machine* - VMs), consolidando várias funções em um único servidor físico. A seguir se encontra descritos alguns conceitos que colaboram com este cenário da virtualização.

2.1.1.1 NFV

O NFV é um conceito de rede que consiste em utilizar técnicas de virtualização de software para provimento de diversos tipos de funções de rede, executando-as em hardware comum de dispositivos de rede, servidores e armazenamento, interligando e criando serviços de comunicação, além de expandir o desempenho nos processos de implementação de novos serviços de rede. Freitas et al. (2018) complementa descrevendo-o como um conjunto de funções com recursos de redes utilizadas para criar ou suportar serviços já utilizados anteriormente na estrutura física, porém virtualizados, promovendo assim uma melhor ordenação e utilização para cada caso.

O NFV facilita o controle de projetos, implementações e novos modelos de administração de serviços de rede, já que permite a possibilidade de separar suas funções, diferenciando *Firewall*, *DNS*, *NAT* (SOARES et al., 2014). Conforme Chiosi et al. (2012) e Han et al. (2015) o NFV utiliza as tecnologias padrão de virtualização para consolidar muitos tipos de equipamentos de rede em servidores, switches e dispositivos de armazenamento de alto volume genéricos.

Ao que antes eram limitações e restrições no hardware que executavam as funções de redes, torna destaque para um sistema virtualizado de rede, visto que permite um melhor controle, garantindo uma maior agilidade e flexibilidade. Segundo Han et al. (2015) a utilização do NFV aumenta o desempenho, o gerenciamento, a segurança, a confiabilidade e a estabilidade. Através de virtualização com NFV, é possível separar as implementações de software das funções de rede da plataforma do hardware, possibilitando um melhor provisionamento para os

serviços de rede.

Segundo Han et al. (2015), as três causas que caracterizam as principais diferenças do NFV para funções em equipamento não virtualizados são: Independência de software com o hardware; inserção automática de funções de redes em recursos de hardware diferentes e o fornecimento de recurso com base no desempenho dinâmico da rede. O gerenciamento da infraestrutura NFV deve ser capaz de manter seus serviços sempre disponíveis, para isto, normalmente as prestadoras de serviços de redes provisionam seus serviços, reduzindo os custos de utilização de recursos devido sua redundância (GREENBERG et al., 2008).

2.1.1.2 VNF

As VNF são funções de rede normalmente executadas em uma ou mais VMs sobre a infraestrutura de rede de hardware, que permite assimilar a execução de programas e processos que eram executados em dispositivos físicos da rede, como roteadores, *switches*, servidores e sistemas de computação em nuvem, normalmente instaladas em servidores de alta capacidade ou infraestrutura de computação em nuvem, gerando assim redes mais rápidas e lucros significativos.

Han et al. (2015) relata que é necessário instanciar as VNFs em pontos corretos da infraestrutura, pois assim, além alocar e dimensionar dinamicamente os recursos de hardware, permite uma maior facilidade para interconectá-los e alcançar o encadeamento de serviços. Segundo Han et al. (2015) ao implantar instâncias de VNF, é necessário projetar algoritmos eficientes que suportem dividir a carga de rede conforme a quantidade de VMs distribuídas pela infraestrutura, analisando sempre os requisitos da latência.

As VNFs podem ser encadeadas ou combinadas em blocos para fornecer serviços de comunicação de rede em escala. Saber explorar as informações obtidas durante a execução das VNFs traz algumas vantagens, como: Disponibilidade e resiliência de falhas; Melhor desempenho de VNFs, comparado com as versões baseadas em hardware correspondentes e Precisão de gerenciamento e no controle de tráfego. Quando comparadas a equipamentos físicos, as VNFs apresentam vários ganhos, relata Chiosi et al. (2012) em seu trabalho.

2.1.1.3 SDN

SDN são redes que utilizam softwares em vez de dispositivos especializados e que normalmente são utilizadas para provisionar e gerenciar serviços de redes e outras aplicações, facilitando a implementação de aplicações programáveis, ampliáveis e sob demanda. (HAN et

al., 2015) define SDN como uma tecnologia de rede que separa o plano de controle do plano de dados subjacente, consolidando as funções de controle em um controlador logicamente centralizado.

Dentro do conceito de SDN, seus recursos podem ser definidos como qualquer coisa que permita ser empregado para fornecer serviços em resposta a solicitações de clientes. Ordonez-Lucena et al. (2017) complementa, explicando que este procedimento inclui recursos de infraestrutura, funções de rede (*Network Functions* - NFs) e serviços de rede. A SDN é uma tecnologia de virtualização que tende a colaborar com o fatiamento de rede, permitindo mais flexibilidade, programabilidade e escalabilidade nas nuvens, relatam (SOARES et al., 2014) e (ZHANG et al., 2017).

A NFV e a SDN são mutuamente benéficas no âmbito das redes 5G, altamente complementares entre si e compartilham a mesma característica de promover inovação, criatividade, abertura e competitividade, além disso, controladores SDN concedem melhor controle das fatias de rede de maneira centralizada.

2.2 Fatiamento de Rede (*Slicing Network*)

Trata-se de uma técnica específica de virtualização, a qual proporciona a implementação de várias redes lógicas sob uma infraestrutura de rede física compartilhada (MADEMANN, 2017). As redes lógicas dos *slices* atuam de ponta a ponta e são formadas por funções de rede, armazenamento e computação (FREITAS et al., 2018).

No fatiamento, a infraestrutura de uma rede é subdividida em partes denominadas *slices*. A *slice* é reservada à uma aplicação de um usuário ou locatário e é considerada como um conjunto de recursos que, combinados com a orquestração, atendem a todos os requisitos de um caso de uso específico (CLAYMAN; TUSA; GALIS, 2018). Para Freitas et al. (2018) o fatiamento de rede é tratado como serviço e cada *slice* é a junção de recursos físicos e/ou virtuais que suportam a inserção de componentes de serviços independentemente de outros *slices*.

A segregação da infraestrutura em múltiplas partes confere benefícios econômicos, tais como: redução de gastos, remoção das limitações de infraestrutura e tecnologias, expansão do cenário para vários clientes diferentes e menor transmissão de dados na rede, isto porque o tráfego de dados móveis é crescente (SILVA et al., 2018).

O conceito de fatiamento de rede permite que provedores de rede compartilhem infraestruturas de maneira flexível com provedores de serviços móveis e verticais (FREITAS et al., 2018).

Um caso de uso que tem como alvo a utilização de nuvens para oferecer serviços e infraestrutura para propósito geral, é composto por três tipos de agentes:

- **Provedor de infraestrutura**: é o provedor que possui e gerencia uma determinada rede física e recursos computacionais, armazenamento e comunicação;
- **Locatário**: aluga os serviços de infraestrutura de rede de um ou mais provedores, sob a forma de rede virtual;
- **Usuário final**: aluga os serviços tanto de infraestrutura de redes ofertados pelos provedores, quanto os serviços fornecidos pelos locatários, porém não consegue fornecer recurso para outros usuários.

A proposta de desempenho da *slice* de rede é entregar uma infraestrutura com melhor latência, robustez, qualidade e velocidade suficiente para cada caso de uso, reduzindo os custos operacionais, como energia e manutenção, e proporcionando o uso mais eficiente de recursos. Para Zhou et al. (2016), o diferencial do uso do fatiamento é a possibilidade de ser criado, programado, executado ou finalizado totalmente sob demanda sem que exista qualquer dependência de um equipamento.

Em relação ao 5G, a utilização do fatiamento de rede física em redes virtuais permite suporte a grande quantidade de casos de uso e aos novos serviços, visto que ambos possuirão diferentes requisitos, podendo haver variação em seus desempenhos. Cabe ao setor de telecomunicações criar soluções de baixo custo que conceda reduzir as variações de desempenhos, como: orquestração total, automação das operações e gerenciamento de fatias de rede.

Nos *slices*, os recursos são unidades gerenciáveis e são divididos em VNFs e recursos de infraestrutura. Para cada *slice* é necessário que exista um VIM gestor do controle dos recursos, suportando diferentes demandas, denominado como VIM *on-Demand* (SILVA et al., 2018). Embora o fatiamento da rede tenha sido amplamente investigado nos últimos anos, ainda carece de discussão no contexto das redes de comunicação 5G (FREITAS et al., 2018).

Para a criação do *Slice* é necessário compreender que o mesmo estará dividido em três fases: Criação, execução e finalização. Para a fase de criação do *slice* é imprescindível a inserção de algumas informações básicas como nome da fatia, proprietário da fatia, dados e período de tempo em que a fatia deve existir e uma descrição dela, o tipo de recursos que irão compor a fatia e suas configurações em termos de capacidade de processamento e memória.

Durante o execução do *slice* é realizado toda transferências de comandos e informações, podendo ser enviadas do VIM com o respectivo serviço e um *feedback* do funcionamento do

mesmo, permitindo assim o locatário analisar as atividades e desempenho. Ao final, após a conclusão da utilização do *slice* é necessário a finalização do mesmo. Durante a etapa de finalização é finalizado e retirado os acessos dos serviços inseridos e realocado os recurso físicos, permitindo o entendimento ao VIM como recursos livres e disponíveis para novas instanciações.

2.2.1 *Slice as a Service*

(ORDONEZ-LUCENA et al., 2017) refere-se o *Slice as a Service* um novo conceito de serviço de rede, cujos recursos são acoplados internamente à *slice* e gerenciados como um todo, podendo ainda conter componentes de serviços independentes de outros *slices*. O *Slice as a Service* possui o conceito de princípio leve, o qual exaure ao limite as configurações dos *slices*, recursos físicos e capacidade de gerenciamento de recursos e desempenho por software, permitindo assim menor tempo de resposta nas alterações das demandas de serviços.

Em uma situação onde os recursos de infraestrutura ficam inutilizados devido ao excesso de recursos, o conceito do *Slice as a Service* torna possível eliminar a ociosidade da rede, criando novos *slices* conforme requisitados pelos clientes, proporcionando diferentes VIMs e funções virtualizadas de rede. Para (FREITAS et al., 2018), os *slices* permitem uma visão geral dos mecanismos, componentes, e abstrações que podem ser utilizados para fornecer um modelo de *Slice as a Service* baseado na instanciação VIM dinâmica.

O problema da alocação dos *slices* possui um forte componente de otimização. Em um sistema com escala maior e com caminhos de rede disjuntos (Figura 3.3), a escolha de enlaces para serem alocados para cada *slice* pode definir a quantidade de clientes suportados pela rede, bem como o desempenho da rede para tais clientes. O VIM é um dos principais componentes de uma estrutura virtualizada, cabe-lhe a responsabilidade de manter o controle e gerenciamento dos recursos de computação, armazenamento e infraestrutura de rede (FREITAS et al., 2018).

Oposto às tradicionais configurações de um único VIM para toda infraestrutura da computação em nuvem, o VIM *on-Demand* é implantado dinamicamente para cada *slice*, cabendo a ele permitir que a *slice* de rede esteja apta ao atendimento de diferentes casos de uso, independentemente da necessidade de execução de qualquer tipo de tecnologia.

2.3 NECOS

O projeto NECOS[1,2] é uma parceria entre universidades brasileiras e europeias com o objetivo de identificar as limitações das infraestruturas de computação em nuvem e, a partir delas, sugerir soluções que respondam às demandas de novos serviços. Sua proposta é o desenvolvimento de uma plataforma que possua facilitadores para o fatiamento de nuvem e que permitam o gerenciamento dos recursos virtualizados de computação, rede e armazenamento de diferentes estruturas (NECOS, 2018) e (SILVA et al., 2018).

O projeto pretende colaborar com o avanço do 5G através de ofertas de *slices* de recursos de forma dinâmica, permitindo eliminar as funções que antes eram executadas diretamente no cliente para as infraestruturas da operadora. Silva et al. (2018) apresentam que para implementação é previsto a utilização do conceito de nuvem definida por fatiamento leve (*Lightweight Slice Defined Cloud* - LSDC), uma abordagem que expande a virtualização para todos os recursos de rede, entre eles as *NFs*, SDN, computação em nuvem, fatiamento de redes, entre outras tecnologias nas áreas de redes de computadores e sistemas distribuídos.

A computação em nuvem é um paradigma já estabelecido para apoiar a implantação de aplicativos e serviços sem a necessidade de contar com infraestruturas onerosas nas instalações. Um provedor de serviços exclusivo às vezes possui suporte à nuvem em seus centros de dados, mas há tendência a uma arquitetura totalmente distribuída, na qual vários provedores terão que cooperar através da interconexão de suas infraestruturas de computação e armazenamento.

A interconexão de uma rede é fundamental no desempenho de um ambiente de nuvem distribuído. Serviços diferentes exigem diferentes recursos de computação e armazenamento, eles também exigem diferentes parâmetros de desempenho relacionados à rede, como largura de banda, atraso ou *jitter*.

O foco do projeto NECOS é o desenvolvimento de novos mecanismos para computação em nuvem, baseado em virtualização de recursos, que permitam o fatiamento de uma nuvem (*cloud slicing*) de maneira leve, sem sobrecarregar a infraestrutura existente; flexível, permitindo sua aplicação em diferentes casos de uso; e gerenciável, possibilitando que cada *slice* possa ser gerenciada de maneira uniforme.

Assim, no contexto do NECOS, a abordagem permite automatizar o processo de configuração ideal da nuvem, estendendo o conceito de virtualização a todos os recursos em um centro de dados, além de fornecer um gerenciamento uniforme com um alto nível de orquestração para re-

[1] intrig.dca.fee.unicamp.br/necos/
[2] http://www.h2020-necos.eu/

cursos de computação, conectividade e armazenamento, atualmente abordados separadamente.

Silva et al. (2018) apresenta as principais características do projeto:

1. Apresentar um novo modelo de serviço – *Slice as a Service*;
2. Habilitar a configuração de *slices* sobre todos os recursos físicos em uma infraestrutura de rede de nuvem;
3. Permitir que cada aspecto que compõe a nuvem, desde a interligação entre VMs até os requisitos de contrato das aplicações hospedadas, sejam gerenciadas por meio de software;
4. Utilizar gerenciamento e virtualização leves (*lightweight*) e uniformes, permitindo a sua implementação em uma variedade de pequenos servidores na nuvens, tanto no núcleo (*core*) quanto na borda (*edge*) da rede.

O NECOS tem como proposta a utilização de uma plataforma de software aberto de última geração. Os principais resultados esperados são:

- Padrões de informações e métodos indispensáveis para especificar um *slice* na qual os recursos e serviços de nuvem fundamentais são detalhados, descobertos e consumidos;
- Instalação de monitoramento leves, flexíveis e em tempo real para federações de nuvem;
- Orquestração em tempo real, permitindo decidir onde, como e quando colocar elementos no nível de *slice* (ou seja, VMs e armazenamento) sobre a nuvem federada;
- Uma plataforma voltada para provedores de serviços em nuvem e provedores de rede que desejam cooperar para fornecer serviços adaptados às diversas necessidades específicas do usuário.

Freitas et al. (2018) comenta que um *tenant* que requisita um *slice* com determinados requisitos de serviço, pode ter sua solicitação atendida ou negada, conforme a disponibilidade de recursos e caso sejam aceitos os requisitos da solicitação de componentes de rede, é analisado e instalado no *slice*.

A plataforma NECOS possui também como premissa a utilização de três bases para seu funcionamento:

- Infraestrutura: Responsável por todos os recursos físicos necessários para o fornecimento dos serviços 5G;

- Orquestração: Responsável pelo gerenciamento do ciclo de vida do *slice* (instanciação, manutenção e finalização);

- Controle de *slice*: Responsável pela criação, alocação dos recursos (computação e armazenamento), implantação do VIM *on-demand* com base na especificação e por manter o controle e gerenciamento dos *slices*.

O *slice* pode ser dividido em dois momentos distintos: Etapa de Criação de *slice* e de Tempo de Execução. A primeira fase é composto por todo processo de criação e disponibilidade do *slice*, é considerado todo processo desde a solicitação do *slice* pelo inquilino, verificação de disponibilidade, configuração de máquinas e instanciação para o inquilino; a segunda fase é um processo sequente, onde as VNFs se encontram criadas em diferentes blocos funcionais dentro de cada *slice* e já operam funcionalmente.

Dentro do projeto NECOS, no modelo de *Slice as a Service*, uma *slice* é constituída por um conjunto de recursos físicos e/ou virtuais (Rede, Computação, Armazenamento) que podem acomodar componentes de serviço sem dependências com outros *slices*. Conforme Freitas et al. (2018) explica, o gerenciamento para infraestruturas prontas para criar *slices*, criam um *slice* no *datacenter* e um *slice* com as funções e serviços de rede sob demanda, após este processo os *slices* são conectados e configurados ou reconfigurados conforme apropriado para fornecer o serviço ao *tenant*.

Os *slices* são criados conforme os tipo de serviço, inclusas a esta, já o VIM *on-demand*, é criado conforme cada *slice*, com finalidade de instalar, controlar e gerenciar os recursos, tornando um parâmetro do *slice*. Outra característica do VIM *on-demand* é a sua implantação na parte superior do *slice*, permitindo incluir versões tradicionais de controladores como Kubernets, OpenVIM e Openstack, sem a necessidade de modificações

Freitas et al. (2018) exemplifica que alguns serviços podem ser melhor atendidos por ferramentas de virtualização de nível inferior, como Xen ou KVM, enquanto outros serão melhor atendidos por plataformas de contêiner, como o Docker ou o Kubernetes. Para uma boa compreensão do funcionamento da plataforma, analise a seguinte situação: Um locatário solicita um *slice* com determinadas propriedades e detalhes de nível de serviço. Essa solicitação é enviada ao orquestrador que mapeia a especificação de *slice* para os recursos de computação, armazenamento e rede e decide quais partes da infraestrutura devem alocar.

Em seguida, o orquestrador entra em contato com todos os controladores de *slice* de infraestrutura e solicita que eles criem uma *slice* em seus domínios. Cada controlador aloca os recursos necessários, instância o VIM *on-demand* para gerenciar os recursos da parte de *slice*

alocado e retorna ao orquestrador os detalhes necessários para acesso ao VIM. Ao final o serviço é implantado no *slice*.

Durante seu ciclo de vida, os *slices* pode aumentar ou diminuir conforme a demanda. Quando os *slices* é finalizada, os recursos são desalocados.

Capítulo 3

ARQUITETURA VIM *on-demand*

Neste capítulo são apresentados os conceitos do VIM *on-demand*. Sua estrutura está dividido no seguinte formato: Seção 3.1 - visão geral, relata a tecnologia 5G e a proposta de implantação do VIM *on-demand*; Seção 3.2 - apresenta o conceito e o funcionamento do VIM; Seção 3.3 - apresenta os conceitos do VIM *on-demand* aponta as principais diferenças com o VIM tradicional. Nas seguintes subseções serão apresentados as seguintes propostas de instanciação do VIM *on-demand*: Subseção 3.3.1 - Instanciação Completa *Bare Metal*; Subseção 3.3.2 - Instanciação Completa com Imagem Pré-Carregada e Subseção 3.3.3 - Instanciação Completa com *Containers*;

3.1 Visão Geral

Com o avanço da tecnologia 5G, visando a integração de diversos serviços para atender numeroso casos de usos, as redes terão de suportar estes diversos serviços sempre com alto desempenho, independente da demanda, utilizando para isso o fatiamento da rede. Freitas et al. (2018) explica que o fatiamento de rede permite aos provedores de rede compartilharem suas infraestrutura de maneira flexível com mercados verticais e outros provedores de serviços.

Ainda neste segmento, o *slice* pode ser representada como uma rede única, fim a fim, assim como para cada uma pode ser implementada uma arquitetura de rede diferente, possibilitando a separação dos locatários na mesma rede operando com o mesmo recurso físico.

O fatiamento de rede permite fazer com que as funções de redes ofereçam um melhor suporte à rede, tornando-a mais flexível, escalável e elástica, além de oferecer maior facilidade na criação das partições lógicas e funcionais da infraestrutura de rede adaptadas para o tipo de serviço a ser fornecido.

O VIM é um dos principais componentes de uma estrutura virtualizada, cabe a ele a responsabilidade de manter o controle e gerenciamento dos recursos de computação, armazenamento e rede da infraestrutura. Ao contrário das tradicionais configurações de um único VIM para toda infraestrutura da computação em nuvem, o VIM *on-demand* é implantado dinamicamente para cada *slice*.

Cabe a ele permitir que a *slice* de rede esteja apta para atender diferentes casos de uso, independente da necessidade de execução de qualquer tipo de tecnologia. O VIM *on-demand* é instanciado para cada *slice* e é responsável por controlar e gerenciar os recursos de um *slice* de maneira automática. O VIM *on-demand* permite um melhor desempenho durante a criação das *slices*, pois, é composto de meios que permitem adiantar os processos de instalação.

3.2 VIM

O VIM é um componente da estrutura arquitetural encarregado do controle e gerenciamento dos recursos de computação, armazenamento e rede da infraestrutura (FREITAS et al., 2018). Atualmente muitos fornecedores de serviços possuem VIM para utilização do seu sistema, como por exemplo a *VMware* com o *vSphere*[1], o *Openstack*[2] que se destaca por oferecer mais funcionalidades do que simplesmente serviços, o *Red Hat*[3], *Mirantis*[4] e o *Oracle*[5] são alguns outros exemplos.

Explorar ao máximo um VIM permite o gerenciamento preciso e, para isso, é proposto que haja uma constante coleta de informações de desempenho e falhas; gerenciamento das imagens dos softwares durante as ações de adição, exclusão, atualização, consulta e copia; e gerenciamento de catálogos de VNFs. É responsável também por aumentar a interoperabilidade dos elementos virtualizados da rede através de blocos funcionais compostos por NFV.

O VIM é fundamental para encontrar vantagens durante ocasiões permitidas pela arquitetura NFV e que, através deles, é criada a coordenação dos recursos físicos necessários para fornecer os serviços de rede, fator que se destaca nos Provedores (*Infraestructure as a Service* - IaaS). Os IaaS assim como o provedores de *Slice as a Service* precisam garantir que os provedores das suas infraestruturas funcionem precisamente e que seus recursos sejam alocados da melhor maneira possível com base nos requisitos de suas redes, servidores, ou armazenamento.

[1] www.vmware.com/br/products/vsphere
[2] www.openstack.org
[3] www.redhat.com
[4] www.mirantis.com
[5] www.oracle.com/

Destaca-se como algumas das principais funcionalidades e responsabilidades de um VIM:

- Descrição detalhada dos recursos virtuais para os recursos físicos através de um catálogo, sempre atualizado, possibilitando um melhor controle de atualização de software, alocação, liberação e recuperação de recursos;
- Gerenciamento do encaminhamento das VNFs, ordenando as redes, sub-redes, portas e demais funções;
- Gerenciamento de um inventário de recursos de hardware e software, mantendo atualizado conforme a descoberta de novos recursos e possibilitando a sua otimização para a utilização dos mesmos.

Para análise de desempenho, observa-se durante o uso que a taxa de transferência e latência será afetada. Essas taxas são um dos principais pontos que apresentam mudanças após a inclusão das VNFs nos sistemas, pois é através desses fatores que é possível uma comparação de desempenho entre a capacidade da VNF em relação a versão física.

Para a inexistência da perda de dados devido a indisponibilidade de serviços na rede, o gerenciador provisiona seus serviços, trabalhando sempre com redundância, evitando aumentos inesperados de tráfego ou falhas. Cabe a ele também gerenciar o controle de elasticidade das fatias para melhorar a utilização dos recursos.

Cabe ao VIM manter o controle de confiabilidade e estabilidade, principalmente ao realizar migração dos dados. Para isso, em situações onde os hardwares passam por uma deformação elástica, é necessário que a plataforma retorne os hardwares utilizados à sua forma original. Após reconfigurar ou reposicionar a sua forma original, a plataforma deve garantir a estabilidade aos demais serviço sem gerar problemas às demais *slices*.

A Figura 3.1 assimila uma arquitetura de alto nível de virtualização e é composta por 3 controladores principais: Orquestrador, Gerenciador VNF e VIM.

- **Orquestrador**: Gerencia e orquestra os recursos de infraestrutura e de software para realizar serviços de rede;
- **Gerenciador VNF**: Trata todos os eventos de origem das VNF como: instanciação, dimensionamento, térmico e atualização;
- **VIM**: Gerencia e virtualiza os recursos configuráveis de computação, rede, armazenamento e interliga com as VNFs.

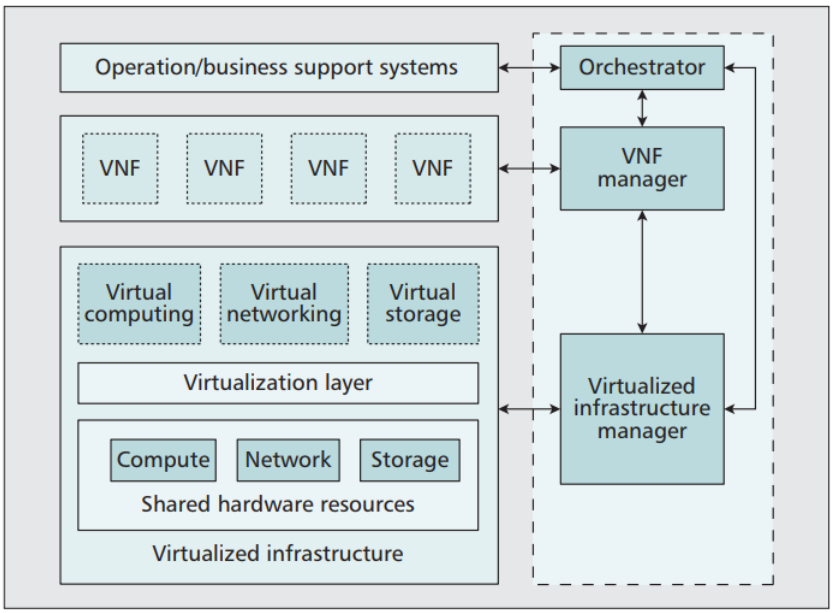


Figura 3.1: Arquitetura NFV.

Fonte: (ORDONEZ-LUCENA et al., 2017).

Compromete-se ao VIM a responsabilidade de criar, finalizar e manter o controle da infraestrutura, realizando toda configuração para instanciação dos serviços requerido na máquina, assim como sua comunicação e seus controladores de serviços através das VNFs de controle. O VIM que antes era único e responsável pela configuração total da computação em nuvem, no conceito do *Slice as a Service* torna-se um VIM *on-demand* no fatiamento de rede e é implantado dinamicamente para cada fatia.

3.3 VIM *on-demand*

O VIM *on-demand* é implantado e configurado dinamicamente para cada fatia conforme suas necessidades de serviço, criando assim redes lógicas fim a fim, executadas em uma determinada rede pressuposta (física ou virtual) com controle e gerenciamento independentes, e isolada das demais redes. Freitas et al. (2018) complementa, relatando que o próprio VIM se torna um parâmetro de fatia e pode ser escolhido e configurado de acordo com as necessidades do serviço. Outra vantagem é a independência da parte que foi instanciada, pois, não é necessário conhecer a abstração da fatia, permitindo que diferentes soluções tradicionais como Openstack, Kubernetes[6], OpenNebula[7] e OpenVIM[8] sejam utilizadas (FREITAS et al., 2018).

[6] www.kubernetes.io
[7] www.opennebula.org
[8] www.sdxcentral.com/projects/openvim/

O gerenciamento pode ser considerado o núcleo do sistema, pois é de sua responsabilidade preparar, manter e finalizar o funcionamento do sistema. Para isso, é necessário instanciar, alocar e dimensionar dinamicamente os recursos nos melhores locais para o locatário, conectando os serviços em uma rede encadeada pronta para qualquer ocasião de flexibilidade de provisionamento de serviços. Nesse caso o gerenciamento deve levar os recursos de uma infraestrutura como um todo, analisando suas variações.

Outra característica do VIM *on-demand* é permitir aumentar a sua capacidade e facilidade de criação de arquitetura com recursos para cada *slice*, na necessidade de possuir sua própria fatia de nuvem auto gerenciável, de ter controle das máquinas com total isolamento e na aplicação de ferramentas de orquestração em recursos gerenciáveis (NECOS, 2018). Outra vantagem é que eles podem ser instanciados na parte superior da *slice* e, como tal, não precisam estar cientes da abstração da *slice* (FREITAS et al., 2018).

Para cada recurso utilizado pelo VIM é alocado um *hypervisor*, onde é feito o gerenciamento de rede, análise de problemas de desempenho e coleta de informações de falhas da infraestrutura. Toda implementação de infraestrutura virtualizada é incluída na camada de virtualização, garantindo assim que tanto as VNFs quanto os recursos configurados sejam independente de hardwares específicos. Esse tipo de funcionalidade é normalmente fornecida na forma de *VMs*.

A etapa denominada como criação de *slice* pode ser demonstrada pela figura 3.2, onde um locatário realiza uma solicitação, passa as especificações e aguarda a chave de acesso final.

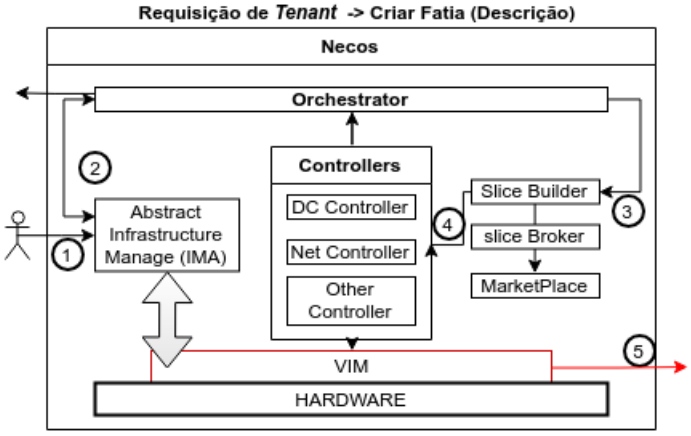


Figura 3.2: Modelo de criação do *Slice as a Service*

Fonte: Elaborado pelo Autor.

A etapa 1, identificada na Figura 3.2 pelo número 1 no centro de um círculo preto, demonstra as informações sendo enviadas para o Gerenciador de Infraestrutura (*Infrastructure &*

Monitoring Abstraction - IMA). Nesse momento o IMA verifica se existe a disponibilidade dos recursos de hardware e software necessários e, caso estejam disponíveis, o orquestrador do *slice* (etapa 2) inicializa o processo de criação do *slice* solicitando ao módulo de criação dos *slices* (*slice builder*). O *slice builder*, por sua vez, verifica qual controlador (*controller*) será utilizado e, após isso, é iniciado o processo de instalação do *controller* no VIM *on-demand*, para que depois o mesmo realize todas instalações na máquina *bare metal* e retorne a chave de acesso ao locatário.

A interface de controle do *slice* (*Slice Controller Interface*) permite que administradores (provedores de rede) e usuários (locatários) interajam com o sistema. Através dessa interface, o administrador pode manter o inventário de recursos do *datacenter* enquanto os *tenants* podem definir solicitações de divisão da rede.

No centro da arquitetura está a camada de serviço de fatiamento que cria a rede virtual, divide e instância os VIMs *on-demand* com base no pedido do locatário. O primeiro componente dessa camada é o *slice Manager*, composto por dois subcomponentes: *User Manager* e *Resource Manager*. O Gerenciador de recursos gerencia todos os recursos no *datacenter* e mantém os recursos e seus estados atualizados.

O componente Gerenciador de usuários gerencia todos os usuários que podem acessar o *DC Slice Controller* nesse site, confirma a identidade de um usuário, verifica suas permissões e impede o acesso inadequado indevido a esses recursos.

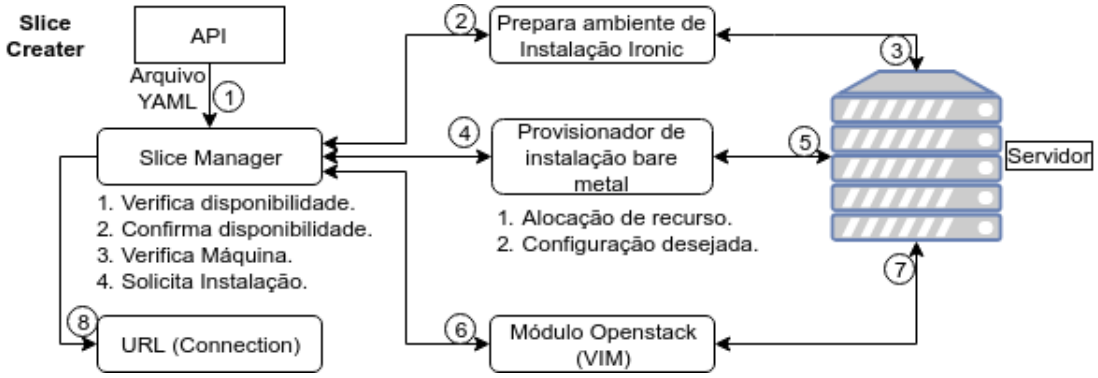


Figura 3.3: Infraestrutura VIM *on-demand*.

Fonte: Elaborado pelo Autor.

O *slice Creator* recebe solicitações de fatia da Interface do Controlador de Fatia e interage com o Gerenciador de Recursos e o Gerenciador de Usuários para determinar se é possível criar uma nova fatia.

A Figura 3.3 apresenta um conjunto de tarefas necessários à implantação de um VIM

on-Demand e permite, conforme a numeração, visualizar este conjunto de tarefas e suas dependências entre si:

1. Chamada de criação de *slice*, em que as informações são inseridas em um arquivo *YAML*, e enviado para o *Slice Manager*.
2. *Slice Manager* solicita permissão para preparar o ambiente, instalando e configurando os pre requisitos, para a instalação *bare metal*.
3. Retorno após ter concluído a preparação do ambiente para instalação *bare metal*.
4. O *Slice Manager* solicita que o provisionador realize instalação, alocação de recurso e configuração desejada via *bare metal*, deixando pronto para a instalação do controlador neste ambiente.
5. Retorno após ter concluído a instalação *bare metal* no servidor de provisionamento.
6. Realiza a instalação do controlador no VIM dentro do servidor. A máquina já se encontra em estado finalizado e pronto para utilização conforme os requisitos necessários inseridos no arquivo *YAML*.
7. Retorno após conclusão de etapa.
8. Retorna a URL e chave de acesso.

É possível ainda diminuir os passos e reduzir o tempo de criação, Figura 3.3, através de outros tipos de instanciação da arquitetura baseado em uso de *containers*, de imagens prontas e auto-configuração. Para que isto aconteça é necessário realizar algumas pré-instalações, permitindo diminuir o tempo de instalação, e, por fim, após já ter um ambiente pronto, basta apenas exportar e importar.

As subseções a seguir apresenta as três propostas de opções de instalação VIM *on-demand*: Instanciação Completa *bare metal*, Instanciação Completa com imagem pré-carregada e Instanciação Completa com *containers*.

3.3.1 Instanciação Completa *Bare Metal*

Uma completa instanciação de um VIM *on-demand* pode ser considerada uma instalação completa (Figura 3.3), exigindo que seja realizado todo processo de instalação, sem procedimentos que reduzam o tempo de instalação, necessitando da execução de todos os passos.

As etapas de instalação são divididas em: Criação de Virtual Machines (*VMs*), Configuração de rede, Comunicação entre *VMs*; Instalação Openstack; Instalação do serviço *bare metal* (Ironic); Provisionamento da máquina *bare metal*; Instalação Kolla-Ansible; Provisionamento de serviço Openstack e retorno da URL de acesso.

A Figura 3.4 detalha os passos e processos durante a execução para a instalação completa, e esta por sua vez, são fracionados em 3 subgrupos: Criação e configuração de *VMs*, Instalação de serviços para o provisionamento *bare metal* e Instalador Openstack (Kolla-Ansible):

1. Chamada *API Creator-Slice*:
    - É realizado a transferência das informações de configuração da máquina requisitado pelo inquilino.
2. Criação e configuração de *VM*:
    - Criar as *VMs*: *Controller* e *Compute*.
    - Configuração de Rede e comunicação.
3. Instalação de serviços para o provisionamento *bare metal*:
    - Serviço Keystone, Glance, Nova, Neutron, Cinder, Swift e Dashboard.
4. Instalação do *Kolla-Ansible* na máquina de provisionamento:
    - Configura e instala conforme os serviços do cliente Openstack.
    - Exporta *snapshot* da máquina de provisionamento (.OVA):
5. Retorna a URL de acesso com a chave credenciada.

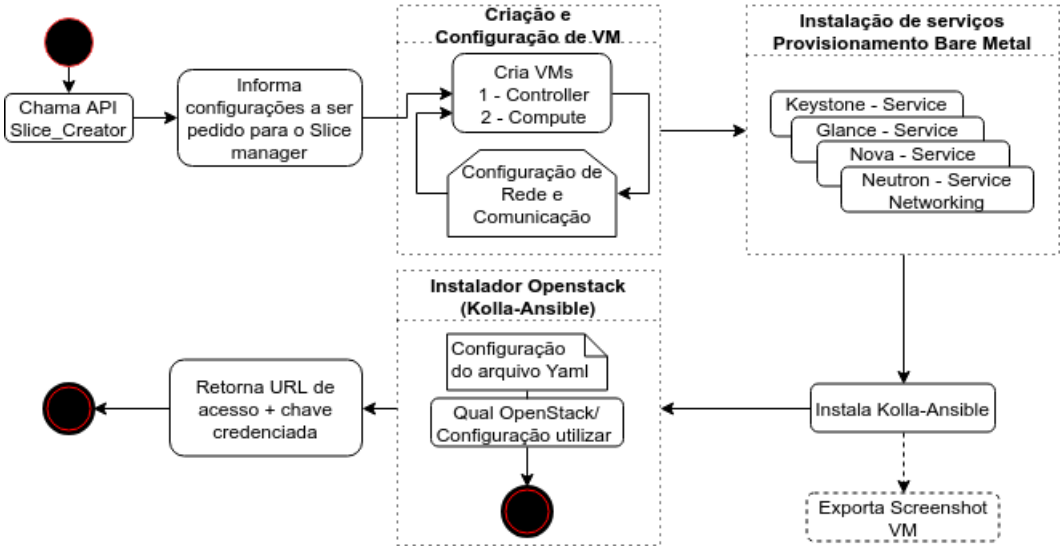


Figura 3.4: ***VIM on-demand Full Deployment.***

Fonte: Elaborado pelo Autor.

Apesar da configuração completa ser a de maior custo de instalação, ela é que proporciona maior poder de configuração de ambiente, e em situações que necessitam de alguma configuração específica e única, este tipo de configuração se destaca.

3.3.2 Instanciação Completa com Imagem pré-carregada

A instanciação de um VIM *on-Demand* com uma imagem pré-carregada, é uma instanciação, na qual parte das etapas encontram-se pré-executadas ou provisionadas da instanciação completa, porém com a máquina em um *snapshot*, necessitando ser importado e reconfigurado. O diagrama representado na Figura 3.5 traz à luz o arcabouço de tarefas compreendidas.

Ao invés de criar uma nova *VM* para instalar os controladores de cada *slice*, realizamos o *import* do *snapshot* da *VM*, que é uma tarefa menos custosa computacionalmente. Com isso, o oneroso trabalho de criação e configuração de *VMs*, instalação e configuração de serviços de provisionamento *bare metal* é evitado, restando apenas a configuração do serviço de rede da nova *VM* e a configuração para as próximas máquinas *bare metal* a serem criadas. As etapas restantes para conclusão de uma instalação completa de uma *slice* com seus serviços são a instalação do kolla-ansible e a instalação e configuração do controlador Openstack que serão utilizados.

A Figura 3.5 apresenta passos necessários para a finalização da instalação. A eliminação do grupo de Criação e Configuração de *VM* e *Install services for bare metal Provisioners* são tarefas que diferenciam os grupos de atividades visualizadas na Figura 3.4, restando o grupo de

instalação Openstack diretamente no Kolla-Ansible.

As próximas etapas a serem realizadas na instanciação com imagem pré-carregada são:

- Importar *snapshot*;
- Verificar comunicação entre máquinas;
- Instalação e configuração do openstack conforme configuração do arquivo *YAML*;
- Retorna URL de acesso com chave credenciada.

Esta instanciação, comparado a instanciação anterior apresenta melhor desempenho devido a etapa de preparação dos dois sub grupos retirados da etapa anterior, criação e configuração de *VM* e instalação dos serviços para provisionamento *bare metal*, (Figura 3.4), nesta instalação pode ser facilmente configurado, e estas serem as etapas que mais consome tempo durante a criação dos *slices*:

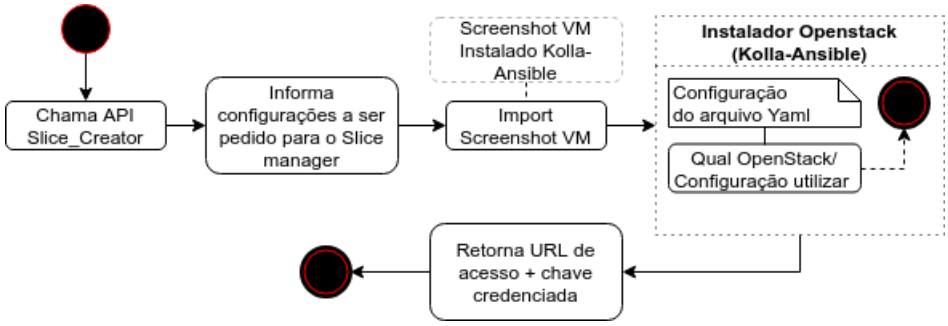


Figura 3.5: VIM *on-demand* pré-carregada.
Fonte: Elaborado pelo Autor.

- Criação de *VM*:
  - Após a importação do *snapshot*, a *VM* pode sofrer alguns erros e necessitar de algumas configurações, como de ip e serviços de Rede e re-execução de alguns serviços.
- *Kolla-Ansible*:
  - Instalação do Openstack *Kolla-Ansible* na máquina importada (*snapshot*).
  - Configuração de acesso com a máquina *bare metal*
  - Instalação do Controlador Openstack conforme os serviços requisitados pelo inquilino.
  - Retorna acesso ao inquilino.

3.3.3 Instanciação Completa com *Containers*

A última instanciação elimina todos os processos de instalação dos subgrupos anteriores: Criação de *VM* base para os serviços, instalação do Openstack Ironic (*Bare Metal*) e instalação do Openstack Kolla-ansible (Controlador Openstack).

Embora as etapas anteriores que pudessem tornar mais confusa e onerosa durante a instalação, esta instanciação simplificada não envolve tarefas de instalação de pacotes ou serviços. Neste método é necessário a importação do *snapshot* pronto, conforme demonstra a Figura 3.6, e configuração.

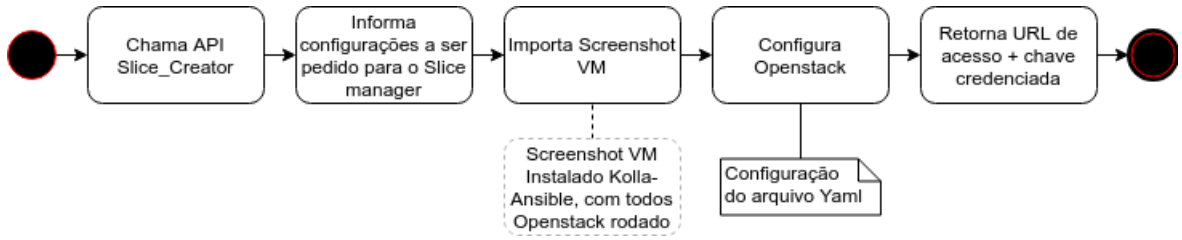


Figura 3.6: VIM *on-demand* com *containers*.

Fonte: Elaborado pelo Autor.

Etapas necessárias para a instanciação completa com *containers*:

- Importação do *snapshot* da *VM*.
- Verificação de comunicação de *VM*:
  - Nesta etapa é necessário realizar uma verificação de comunicação entre máquinas para conferir se o sistema está realmente em funcionamento.
- Configuração de acesso com a máquina *bare metal*
- Instalação do Controlador Openstack conforme os serviços requisitados pelo inquilino.
- Retorna acesso ao inquilino.

Capítulo 4

TRABALHOS RELACIONADOS

4.1 Trabalhos Acadêmicos

Apesar do conceito de 5G ser considerado novo, existem diversos trabalhos acadêmicos que colaboram com o seu desenvolvimento, porém o único que utiliza o conceito de *Slice as a Service*, fator central para o provisionamento VIM *on-demand*, é o projeto NECOS[1], no qual este trabalho faz parte (SILVA et al., 2018).

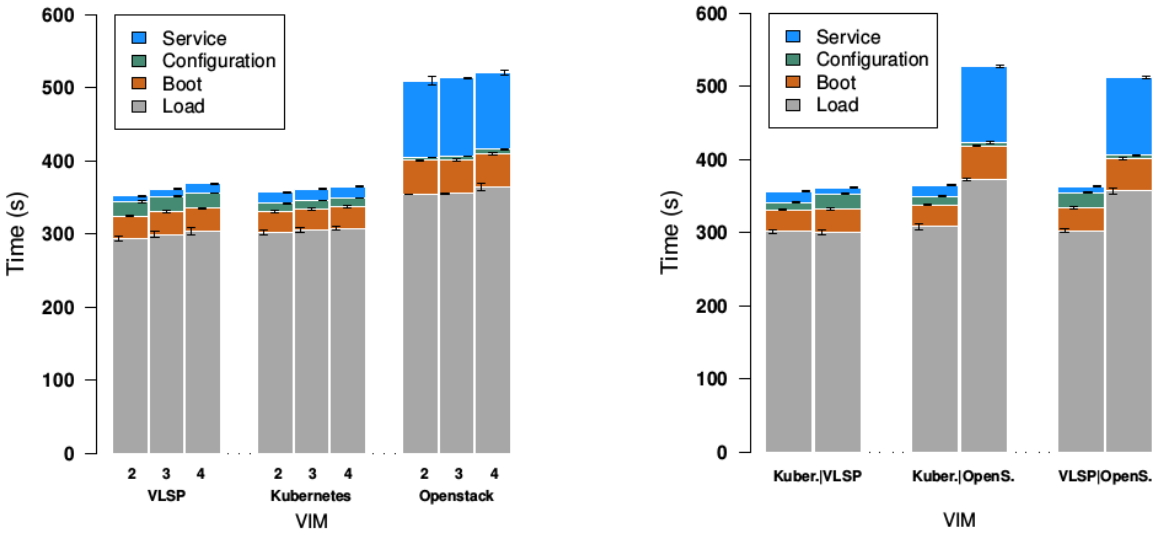
Como outros projetos e trabalhos acadêmicos, destacam o 3GPP, o qual apresentou a proposta de padronização do sistema de rede de acesso 5G e do núcleo de sistema de rede 5G ao mesmo tempo (KIM; KIM; CHOI, 2017); o 5G NORMA, que propôs-se uma arquitetura capaz de lidar eficientemente com diversos requisitos e flutuações na procura de tráfego resultantes de conjunto de serviços heterogêneos e suspensos (ROST et al., 2017); e o SONATA abordou os desafios significativos associados ao desenvolvimento e à implantação dos serviços complexos previstos para as redes 5G (DRÄXLER et al., 2017).

O trabalho (FREITAS et al., 2018), apresenta um sistema capaz de implantar e fornecer VIMs operacionais completos através de um *Data Center (DC) slice Controller*, com base em uma arquitetura em que as fatias do DC são criadas sobre recursos transformáveis (computação e armazenamento).

A arquitetura foi dividida em três camadas: a interface do controlador de fatia (*slice Controller Interface*), a camada de serviço de fatiamento (*Slicing Service Layer*) e a camada de recurso (Infraestrutura): O *slice Controller Interface* é responsável pela interação do sistema com os provedores de serviço e locatários/inquilinos. O *slice Creator* recebe os pedidos de *slices* e comunica com o *slice Manager* para verificar se é possível criar o *slice*. Caso seja possível, o

[1] http://www.h2020-necos.eu/.

slice Creator iniciará o procedimento de registro da nova fatia (e recursos) no Armazenamento de Informações do *slice* e entrará em contato com o *VIM*, que por sua vez é configurado para usar os recursos que foram escolhidos pelo *slice Creator*.



(a) Tempos discriminados por VIM e número de nós utilizados.

(b) Tempos discriminados por conjunto de VIMs usados nos testes.

Figura 4.1: Resultados do *DC Slice Controller*.

Fonte: (FREITAS et al., 2018).

Durantes os seus testes foram utilizadas imagens pré-configuradas para três *VIMs* distintos: VLSP, Kubernetes e Openstack, sendo realizado em duas etapas, sendo o primeiro com testes dos *VIMs* em particular e o segundo em duplas. Conforme a Figura 4.1 as analises verificadas durantes os testes foram:

- Tempo de Carregamento: o tempo necessário para carregar a imagem VIM nos nós;
- Tempo de Inicialização: o tempo necessário para iniciar o sistema operacional após a aplicação da imagem;
- Tempo de Configuração: no qual são realizadas as configurações necessárias para iniciar o VIM;
- Tempo de Inicialização do Serviço: que representa o tempo para o VIM estar em execução após a configuração.

Freitas et al. (2018) relata que o Openstack, possui maior tempo devido a necessidade de configuração e possui maior tamanho (imagem) devido ser composto de módulos diferentes,

necessitando da inicialização de todos os serviços (Compute, Keystone, SQLDB, RabbitMQ e Neutron).

O presente trabalho tem como diferenças do trabalho de (FREITAS et al., 2018), a instalação total do ambiente de controle sob demanda analisando diferentes abordagens para carregamento do ambiente. Além disso, entende-se que ao realizar a instalação do ambiente ao invés de importar imagens prontas, mais flexibilidade e opções de customização de tal ambiente podem ser exploradas. Outro fator que permite aumentar o desempenho quando comparada com o trabalho, é em relação a proposta de Instalação Completa com *Containers* (Subseção 3.3.3) que permite eliminar o passo da instalação pré-carregada, carregamento da imagem, melhorando o desempenho durante a instalação.

4.2 Trabalhos Industriais

Existem diversos trabalhos voltados à indústria, e para o presente trabalho foram analisados o Kayobe[2], que é um projeto Openstack com finalidade de automatizar a implantação do Openstack em um conjunto de servidores *bare metal*, utilizando *containers*; e o TripleO (Openstack On Openstack)[3] que é um projeto do Openstack que permite implantar duas nuvens, com a finalidade de instalar, atualizar e operar nuvens Openstack utilizando as próprias instalações de nuvem do Openstack como base.

Em relação à característica inovadora nessa criação de *slice*, comparado com a utilização de recursos no OpenStack Nova, compreende-se que o mesmo permite assimilar uma mini-nuvem para o locatário.

O *Kayobe*[4] é um projeto Openstack inicialmente desenvolvido por StackHPC[5] com finalidade de automatizar a implantação do Openstack em um conjunto de servidores *bare metal*, utilizando *containers* (KAYOBE, 2018), ele é composto de *playbooks Ansible* e um módulo python, tendo estes como objetivo complementar o projeto da *Kolla-Ansible*, fornecendo uma implementação do Openstack altamente personalizável e configurável, além da automação de muitos procedimentos operacionais.

Utilizar os *containers* com os serviços do Openstack permite aumentar o isolamento; porém, executar um plano de controle em um orquestrador, como o Kubernetes ou o Docker Swarm, aumenta os custos operacionais e sua complexidade de maneira significativa. Os *hosts* em um

[2]https://kayobe.readthedocs.io.
[3]https://wiki.openstack.org/wiki/TripleO.
[4]https://kayobe.readthedocs.io
[5]https://www.stackhpc.com

plano de controle do Openstack devem, de alguma forma, ser provisionados, mas implantar uma nuvem Openstack secundária para fazer isso parece um exagero. Apesar de ser um projeto inovador e que pode acrescentar muito na indústria, o *kayobe* necessita imprescindivelmente dos seguintes serviços:

- Openstack *Bifrost*: Descobrir e provisionar a nuvem;
- Openstack *Kolla*: Construir imagens de contêiner para serviços Openstack.

O *TripleO* (Openstack On Openstack)[6] é um projeto do Openstack para implantar duas nuvens, com a finalidade de instalar, atualizar e operar nuvens Openstack utilizando as próprias instalações de nuvem do Openstack como base, permitindo assim, através de serviços dos seus serviços automatizar o gerenciamento de frotas em escala de *datacenters* (TRIPLEO, 2018).

O objetivo principal do projeto é permitir, de forma acessível, a implantação e o gerenciamento de uma nuvem de produção em hardware *bare metal* usando um subconjunto de componentes existentes do Openstack. Lemoine (2018) explica que o *TripleO* é favorável em serviços reais, onde os operadores precisam gerenciar operações complexas, como a implementação de componentes do Openstack em todo o ciclo de vida do ambiente físico, pois ele usa um base (Nuvem) Openstack para criar uma infraestrutura de *Subcloud* para o Openstack chamado *Overcloud*.

A Figura 4.2 exemplifica uma situação atual, o serviço Nova não pode executar dois *hypervisors* diferentes em uma nuvem. Para solucionar isso o *TripleO* executa mais duas nuvens acima de outra nuvem:

- A nuvem base é uma nuvem *bare metal* que é executada e constituída de todo o hardware.
- As nuvens acima são nuvens comuns baseada em VMs, sendo executada como locatário/inquilino na nuvem *bare metal*.
- Nuvens de VM adicionais podem ser executadas como locatários paralelos na nuvem subjacente.

[6]https://wiki.openstack.org/wiki/TripleO

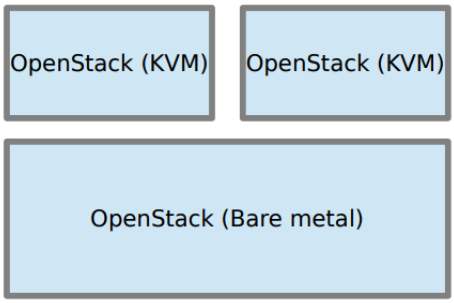


Figura 4.2: Nuvem sobre Nuvem.
Fonte: Openstack TripleO (TRIPLEO, 2018).

A implantação de *TripleO* consiste em três fases principais:

1. Os componentes do Openstack base são instalados em um nó de controle da Nuvem Base.
2. Os componentes do Openstack *Ironic* da Nuvem Base implantam automaticamente as imagens do Sistema Operacional pré-configurado nos Nós Sob Nuvem selecionada.
3. O Openstack *Heat* orquestra a configuração do serviço Openstack de sobreposição nos Nós Sob Nuvem. Outra implantação manual pós configuração do serviço Openstack ainda pode ser necessária posteriormente.

Em cada fase de implantação, o *TripleO* utiliza os Modelos de Orquestração de *Heat* (HOT) que devem ser preenchidos em coerência com os componentes de rede que interconectam os servidores ou as VMs onde os nós das Nuvens Acima devem ser implantados, explica (LEMOINE, 2018).

Em relação à característica inovadora nessa criação de *slice*, comparado com a utilização de recursos no OpenStack Nova, compreende-se que o mesmo permite assimilar uma mini-nuvem para o locatário.

Capítulo 5

IMPLEMENTAÇÃO

Para realizar a implantação do ambiente foi utilizado o software de virtualização Virtualbox em conjunto com Vagrant, os quais são softwares com finalidade de criação e manutenção de ambientes de desenvolvimento virtualizados . A fim de controlar as máquinas virtuais em nível físico foi utilizado o Virtual BMC.

Com o objetivo de gerenciamento e provisionamento das máquinas físicas foi utilizando o Openstack Ironic. Para instalação de controladores que permitam operar nuvens utilizamos o Openstack Kolla-Ansible, pois este é composto por *containers* prontos para produção e ferramentas de implantação de Openstack, facilitando e reduzindo o tempo de instanciação dos serviços.

Este capítulo esta divididos em seções na seguinte sequência: Seção 5.1 - Tecnologias, apresentando as tecnologias e suas devidas versões utilizadas; Seção 5.2 - Ambiente de Implementação, relatando as configurações do ambiente utilizando; Seção 5.3 - Implementação, explicando quais foram os passos e etapas durante o desenvolvimento; e Seção 5.4 - Experimentos e Resultados, relatando quais foram os experimentos realizados.

5.1 Tecnologias

Nas próximas subseções estão descritos os detalhes de cada tecnologia utilizada, sendo estes divididos em seções das tecnologias de virtualização e dos serviços do Openstack.

5.1.1 Tecnologias de Virtualização

5.1.1.1 Virtualbox

O VirtualBox é um software de virtualização que permite, atráves de um *hypervisor*, criar ambientes para instalação diferentes sistemas operacionais como: Windows, linux, macOS, Solaris. Ele simula uma máquina com seus recursos físicos e permite a instalação e utilização de um sistema operacional sobre outro, assim como seus recursos, softwares, porém utilizando um hardware fisicamente compartilhado. Este software é utilizado tanto em ocasiões simples, como por usuários comuns, quanto em situações mais complexas como, desenvolvimento de sistemas ou virtualização de equipamentos de TI.

Outra característica é a permissão de controle de máquinas a partir de linha de comando ou possivelmente remotamente. O Virtualbox também permite a exportação de máquinas criada para uso posterior, assim como suas configurações que são armazenadas em XML e separada das máquinas locais, possibilitando ser transferidas e utilizadas em computadores diferentes.

5.1.1.2 Vagrant

A ferramenta Vagrant é composto por um sistema automatizado que reduz o período de criação, configuração e gerenciamento de ambientes de desenvolvimento de software virtual. o Vagrant suporta trabalhar com cinco softwares de virtualização: VirtualBox, Hyper-V, *containers* do Docker, VMware e AWS. Uma das suas facilidades é a instanciação de sistemas através do download de box, criando assim maior facilidade durante as configurações dos ambientes de trabalho, de maneira portátil e reproduzível, proporcionando aumentar a produtividade aos utilitários deste.

5.1.1.3 Virtual BMC

O Virtual BMC é um pequeno cliente que permite aos usuários criar um virtual BMC para gerenciar máquinas virtuais utilizando o protocolo IPMI, semelhante ao gerenciamento de máquinas de *bare metal* reais. Sua instalação é dada pelo comando pip, e suas principais funcionalidades de controle de máquina virtual são: Ligar e Desligar; Verificar Status de energia; Configurar os dispositivo de inicialização de rede, hd ou cdrom; e Capturar o dispositivo de inicialização atual da máquina virtual.

5.1.2 Tecnologias de Gerenciamento de Nuvem

5.1.2.1 Openstack

O Openstack[1] é um software capaz de gerenciar os componentes de diversas infraestruturas virtualizadas (FIFIELD et al., 2014). É considerado um sistema operacional para nuvem que permite ser implantado tanto em pequena, quanto em larga escala, e possui a capacidade de controlar todos os recursos disponíveis na oferta dessa infraestrutura: máquinas virtuais, rede, armazenadores e balanceadores de carga.

O Openstack é constituído por cinco (5) serviços, sendo três (3) considerados como principais (Figura 5.1), e cada serviço possui suas próprias funções e funcionalidades específicas (SEFRAOUI; AISSAOUI; ELEULDJ, 2012):

- Infraestrutura Computacional (Nova[2]): Plataforma de gerenciamento que controla a infraestrutura, permite gerenciar grandes redes de VMs e arquiteturas redundantes e escalonáveis. Fornece também uma interface administrativa e uma *API Server* (Nova-API) para controle do Openstack. O Nova permite realizar a orquestração da nuvem, incluindo gerenciamento de servidores, redes e controle de acesso.
- Infraestrutura de Armazenamento (Swift[3]): Responsável por criar um espaço de armazenamento redundante e escalonável para armazenar os dados;
- Gerenciamento de Imagens (Glance[4]): Fornece serviços de armazenamento, registrando e distribuindo as imagens para os discos da máquina virtual;
- Outros serviços:
  - *Dashboard* do Openstack (Horizon[5]);
  - Keystone[6]: Serviço de Identidade e autenticação do Openstack.

[1] https://www.openstack.org/
[2] https://docs.openstack.org/nova/
[3] https://docs.openstack.org/swift/
[4] https://docs.openstack.org/glance/
[5] https://docs.openstack.org/horizon
[6] https://docs.openstack.org/keystone

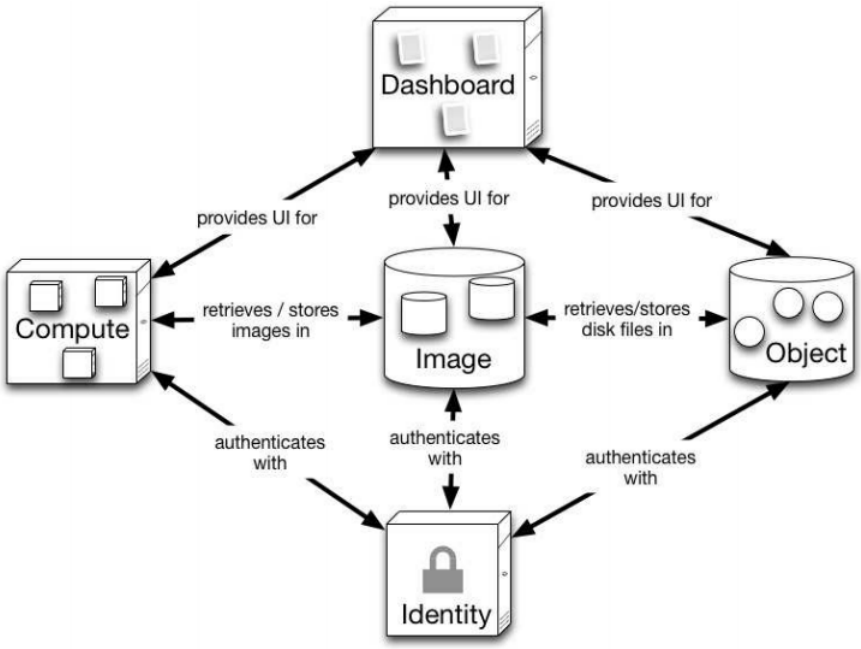


Figura 5.1: Arquitetura Openstack.

Fonte: (SEFRAOUI; AISSAOUI; ELEULDJ, 2012).

O Openstack pode gerenciar máquinas físicas quanto em máquinas virtuais, o tornando um dos principais diferenciais de outras tecnologias de gerenciamento de infraestruturas. Os recursos apresentados pelo Openstack permite um gerenciamento do ciclo de vida das instâncias de máquinas virtuais, assim como seus recursos computacionais, de rede e físico. Seu controle é realizado através de API REST, que por sua vez permite ativar instâncias em servidor, criar imagens, *containers* e objetos de armazenamento e finalizar ações na nuvem do Openstack.

Freitas et al. (2018) relata que quando igualado outros controladores como VLSP e Kubernet, demora mais para instanciar devido ser composto de vários módulos diferentes que precisam das inicialização de todos os serviços, e devido levar mais tempo para sua configuração, devido ser mais complexo e envolver configurar o IP estático e o nome do *host* dos nós, alterar os arquivos de configuração com novos IPs, atualizar a lista de Nós de Cálculo e alterar a senha de administrador do Openstack.

O Openstack tem diversas versões e maneiras de ser instalado em um sistema, como: Openstack-Ansible[7], Mirantis Fuel [8], Packstack[9] e instalação em pacotes da RedHat Stack[10], cada um possuindo o seu propio diferencial durante o processo de instalação:

O Openstack-Ansible realiza a instalação dos controladores Openstack através de funções

[7]https://docs.openstack.org/openstack-ansible
[8]https://www.mirantis.com/software/openstack/fuel/
[9]https://www.rdoproject.org/install/packstack/
[10]https://www.redhat.com/openstack

Ansible para a implantação e configuração de um ambiente. O Mirantis Fuel é um projeto de gerenciamento Openstack em conjunto com o StackLight e é composto por um conjunto de ferramentas de registro, monitoramento e alerta, e permite facilitar a implantação e execução de vários tipos de configurações do Openstack em escala, evitando bloqueio de fornecedores, e acelerando o processo complexo e sujeito a erros.

O Packstack possibilita fácil criação de uma nuvem como prova de conceito em um nó, usando o seu utilitário próprio de instalação, composto por módulos *Puppet* que permite inserir várias partes do Openstack em servidores pré-instalados e já com *SSH*. A plataforma de nuvem do Red Hat Openstack combina suas funções do Red Hat Enterprise com a tecnologia do Red Hat Openstack, com finalidade de criar e gerenciar nuvens mais seguras e escalonáveis sendo tanto públicas quanto privadas.

Apesar do Openstack possuir diversos instaladores que permitem um controle de nuvem com suas próprias características, os mesmos ainda não atuam sob a demanda do cliente.

5.1.2.2 Openstack Ironic

O Openstack *Ironic*, é um versão do Openstack que permite gerenciar e provisionar máquinas físicas, e é composto pelo serviço Nova-Compute e um conjunto de APIs que interagem com os *hypervisors* de *bare metal*. As principais interações de serviço do Openstack com os serviços *bare metal* são: Keystone, glance e neutron. Para funcionamento correto do Openstack *Ironic* e provisionamento *bare metal* é necessário que alguns pré-requisitos sejam atendidos, como:

- Pacotes dependentes para configurar um nós de serviço *bare metal*, como o tftp-server, ipmi, syslinux, etc.
- Nova deve ser configurado com permissão de utilização do terminal de serviço *bare metal*.
- As imagens devem estar disponibilidades e configuradas para acesso no Glance.
- Os *drivers* da máquina Computer deve estar configurado para usar o *driver Ironic* no(s) nó(s) do serviço Nova-Compute.
- Hardware para ser realizado o procedimento através do serviço RESTful da API *Ironic* .

O *Ironic* utiliza por padrão o PXE e o IPMI para provisionar e ligar/desligar as máquinas.

5.1.2.3 Ansible

O Ansible é uma ferramenta de automatização com finalidade de instanciar e gerenciar diversas máquinas ao mesmo momento através de *playbooks* ansibles. Os *Playbooks* são arquivos contendo código com comandos para execução do Ansible, escrito no formato YAML, e permite escrever um conjunto de tarefas de uma única vez, como as etapas de execução para uma determinada máquina. Sua conexão é por acesso SSH para se acessar os demais servidores e executar as *tasks*. A estrutura do arquivo YAML é composto por:

- *Playbooks*: Apresenta os passos para os processos de configurações.
- *Play*: Contem as *Tasks* e define as propriedades de utilização delas, podendo ser portas, nomes de *hosts*, permissão de acesso.
- *Task*: Definições de onde será realizado o trabalho, como o que será instalado ou para qual localização mover determinado arquivo.
- *Module*: Realiza realmente o trabalho de execução.

5.1.2.4 Openstack Kolla-Ansible

O Openstack Kolla-Ansible é uma junção dos Openstack Kolla com o conceito do Ansible. O Openstack Kolla apresenta um conjunto de *containers* prontos e composto por ferramentas de implantação que auxilia com o gerenciamento de nuvem Openstack. O Kolla-Ansible permite uma configuração completa dos *containers* através de configuração Ansible, aumentando a facilidade de implantação do OpenStack. Utilizar o Ansible em conjunto com o Openstack permite uma maior interação do responsável pelo provisionamento, pois conforme aumenta a experiência, mais fácil se torna alterar as configuração do Openstack de acordo com os requisitos do operador.

Assim como os demais serviços Openstack, necessita dos serviços básicos para sua instalação e funcionamento. Normalmente é composto pelas máquinas denominadas *controller* e *computer* que recebem os principais serviços de atuação, e um nó onde é realizado toda implementação de serviços Openstack. Como é composto por diversos *containers* prontos para produção e ferramentas de implantação para operar nuvens Openstack, após o preparo do seu ambiente, torna-se fácil o provisionamento do seu serviço.

Quando comparado a metodologia de criação de *slices* proposto neste trabalho com o Openstack Nova, entende-se que para a instalação, instanciação, controle e finalização utili-

zando a plataforma de gerenciamento Nova é mais complexa, visto que exige todo esforço passo a passo, sem a existência de redução de processos, entretanto, a aplicação proposta inclua a junção do Openstack tradicional com o Openstack *Kolla-Ansible*, este por sua vez vem com os serviços previamente inseridos em *containers*, necessitando apenas da instalação dos serviços básicos para funcionamento.

O principal motivo de não instalação apenas do *kolla-ansible* é que o mesmo não permite instanciação do serviço *bare metal* devido não dar mais assistência para este serviço, exigindo a necessidade de instalação total do Openstack para instanciação do mesmo. Outro fator de diferencial do mesmo é o controle de versão para a instalação, visto que existem diversas versões e que para cada exige uma metodologia de instalação vertical.

O Openstack também permite se tornar um sistema *multi-thread* ou seja executar longas tarefas no mesmo momento, para isto é necessário a implantação de duas bibliotecas do python: *eventlet* e *greenlet*. Para atender múltiplos pedidos simultaneamente é necessário escalonar o Openstack, para isto é preciso alterar as chamadas de serviços pela API, os grupos de segurança, dividir o banco de dados e as filas de mensagens para que seja possível rastrear o estado dos fractais e coordenar a comunicação entre os serviços.

Apesar das tecnologias apresentadas colaborarem com o desenvolvimento do *slice as a service*, ainda é difícil concretizar ao projeto final dado como proposta pelo projeto NECOS (Seção 2.3). Uma das dificuldades ainda imposta é o *stiching* dos serviços instalados com o resto da infraestrutura, visto que para cada caso de uso deve ser considerado como um *slice*, podendo cada desejar o seu próprio VIM, versão, comunicação, serviço, e até a comunicação entre dois *slices* diferentes, necessitando de uma comunicação das máquinas *bare metal* com a rede via a criação de regras através do *Openflow*.

Apesar da atualização em servidores e hardwares físicos serem menos frequentes, ainda é um dos principais empecilhos e dificuldades, visto que com o *slice* os seus dados, serviços e controlador deve-se manter isolado dos demais, aumenta a necessidade de maior confiabilidade e segurança. Outros fatores que devem ser analisados são migração dos dados, duplicidade para sempre mantê-los disponíveis e tempo de inatividade.

5.2 Aparato Computacional

Para criação e controle das máquinas virtuais foi utilizado o software Vagrant, versão 2.2.3. Para exportação e importação do ambiente virtualizado foi utilizado o software Virtualbox, versão 5.2.18. Apesar de ambos permitirem criar, gerenciar e executar uma ou mais máquinas virtuais (*VMs*) tanto isoladamente quanto simultaneamente, a utilização do Vagrant deu-se devido às suas características de fácil controle de *VMs*. Para a implementação do VIM *on-Demand* foi utilizado um servidor com as especificações descritos na Tabela 5.1.

Tabela 5.1: Quadro de configurações do servidor utilizado.

Componente	Dimensão
Processador	i5 3.30GHz 12 núcleos
Disco Rígido	1 terabytes
Memória RAM:	32 Gigabytes
Sistema Operacional	Linux Ubuntu Server 64 bit

As configurações para criação das *VMs* devem ser descritas no arquivo Vagrantfile. Utilizou o Virtualbox durante o processo de exportação e importação dos *snapshot*. As máquinas virtuais denominadas de *controller* e *compute* possuem as mesmas configurações. O Virtual BMC servi-se para gerenciar e monitorar as VMs a nível de máquinas *bare metal* reais.

Com o aparato computacional preparado, o processo de criação do VIM *on-demand* foi dividido em três partes: *bare metal*, Kolla-Ansible e a Versão do *Controller* Openstack. A execução ordenadas das etapas representa apenas a instanciação completa *bare metal*.

5.3 Descrição dos experimentos

Para atender aos requisitos do ambiente proposto, definiu-se um modelo de arquitetura conforme apresentada na Figura 5.2, onde é possível visualizar as etapas de execução da implantação das tecnologias utilizadas em conjunto com as tarefas de importação e exportação de imagens. É possível observar que as atividades de instalações e configurações foram divididas em 3 ambientes, sejam eles:

- ***Machine Server*:** Máquina Física que recebe a instalação de um sistema e dos softwares responsáveis por criar e controlar um ambiente virtualizado;
- ***Machine Bare Metal*:** Máquina Virtual composta pelos serviços Openstack, e responsável pela instanciação da Máquina Física *Bare Metal*;

- ***Machine Full*:** Máquina Virtual anterior composta também com o Kolla-Ansible e responsável pela instalação dos serviços Openstack na Máquina Física *Bare Metal*.

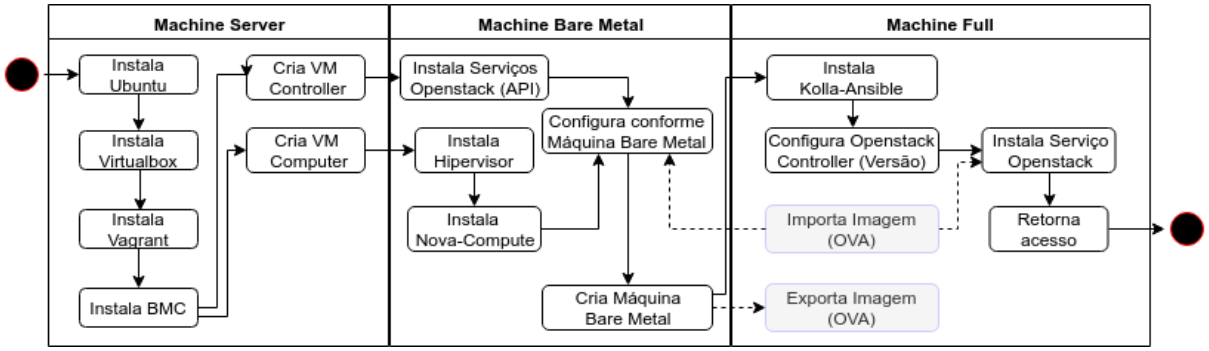


Figura 5.2: Diagrama de atividades em conjunto com as tecnologias utilizadas em cada VM.
Fonte: Elaborado pelo Autor.

Para a exportação e importação das imagens durante a instanciação com imagem pré-carregada e instanciação completa com *containers*, utilizou-se o comando VBoxManager, o qual é uma interface de linha de comando utilizada para controle do VirtualBox, que permite acessar todos os recursos presentes na interface gráfica do usuário.

Para atribuição de novo *slices*, desalocação e controle de recursos são através de chamadas *REST*. A seguir são apresentado as chamadas *REST* utilizadas para ativar as principais funcionalidades com suas respectivas saídas:

1. *Criar um slice*: É realizado uma chamada através do *POST* com algumas informações do *slice*.

```
curl -X POST -d '{"cpu":"2", "mem":"4", "name":"TestOpenstack",
                  "vim-on-demand": "on", "vim-type": "openstack",
                  "vim-style": "bare-full"}'
                  http://localhost:7080/slice/
```

   O retorno é uma resposta contendo o ID do *slice* e alguns detalhes do VIM:

```
{"message":"create-slice","payload": { "id":1123852145, "cpu":"2",
 "mem":"4", "vim-type":"openstack", "vim": {"hostname":"TestOpenstack",
 "port":8888}}, "timestamp":1124135160142}
```

2. *Excluir um slice*: É realizado uma chamada de *DELETE*, passando a ID, que foi retornada quando criou o *slice*.

```
curl -X DELETE http://localhost:7080/slice/1123852145
```

O retorno é uma resposta contendo o ID do *slice*:

```
{"message":"delete-slice","payload {"id":1123852145},
"timestamp":1124135160142}
```

A execução do experimentos contemplaram 27 repetições para cada aplicação de VIM *on-Demand* proposto, utilizando cálculos de tomada de tempo para verificação de desempenhos durante sua instanciação. Cada execução completa foi dividida em partes e os seus tempos de execução foram capturados. A execução completa para criação de um VIM *on-Demand* foi subdividida em:

- ***BM***: Representa o tempo dos procedimentos de criação e configuração de VMs (Controller e Computer), instalação do Openstack Ironic, configuração e instanciação da máquina *bare metal*;
- ***KA*** (Kolla-Ansible): Representa o tempo dos procedimentos inicializados após a finalização da criação da máquina *bare metal*, a instalação do Openstack Kolla-Ansible e configuração de ambiente para instalação do controlador Openstack na máquina *bare metal*;
- ***OS*** (Openstack): Representa o tempo do processo de configuração e instanciação da máquina para instalação do serviço Openstack;
- ***TOTAL***: Combinação dos períodos do *Bare Metal* (BAREMETAL) + Kolla-ansible (KA) + Openstack (OS);
- ***ExpVMBM*** e ***ExpVMKA*** (Export VM): Tempo correspondente a exportação de máquina virtual;
- ***ImpVMBM*** e ***ImpVMKA*** (Import VM): Tempo correspondente a importação da máquina virtual;

Tabela 5.2: Configuração das Máquinas Virtuais

CPU/Mem	4 GB	9 GB	14 GB
2 Núcleos	2N4G	2N9G	2N14G
4 Núcleos	4N4G	4N9G	4N14G
6 Núcleos	6N4G	6N9G	6N14G

Uma execução completa pode ser considerada como a execução de todos os procedimentos de instalação e a configuração até o resultado final de criação da fatia com o serviço Openstack Tacker. Foram realizados três testes (execuções completas) para cada etapa da criação do *slice*. As configurações das VMs foram separadas por núcleos e memórias, sendo criadas conforme a Tabela 5.2.

5.4 Discussão dos Resultados

Após a captura do tempo de cada etapa durante o processamento, visando realizar cálculos de tomada de tempo para verificação de desempenhos, extraiu-se os menores valores de tempo compilados na Tabela 5.4. Observando os valores dos tempos para cada configuração de máquina virtual, conclui-se que o melhor computo de tempo foi 3,65 minutos, se refere ao teste ***OS*** seguido do teste ***KA*** que computou 9,52 minutos. A Tabela 5.3 apresenta a média das execuções de cada configuração de máquina, onde é possível observar novamente que as etapas ***OS*** e ***KA*** computaram os menores tempos médios, respectivamente 4,10 e 11,50 minutos. É possível observar também que as etapas ***KA*** e ***OS*** sucedem em ordem de tempo, tanto para a tomada de menor tempo quanto para a de tempo médio.

Tabela 5.3: Média dos Tempos (Min)

VM	BM	KA	OS	TOTAL
2N 4G	67,84	20,10	5,19	94,27
2N 9G	59,31	14,85	4,33	78,95
2N 14G	47,83	10,35	3,99	62,51
4N 4G	50,93	18,29	4,77	73,80
4N 9G	42,82	11,50	4,10	58,51
4N 14G	40,97	10,35	3,74	55,46
6N 4G	46,07	14,62	4,46	64,84
6N 9G	40,84	10,87	3,99	55,38
6N 14G	39,27	9,52	3,65	52,62
Média	46,07	11,50	4,10	62,51

Tabela 5.4: Menores Tempos (Min)

VM	BM	KA	OS	TOTAL
2N 4G	58,89	20,10	5,19	94,27
2N 9G	56,61	14,85	4,33	78,95
2N 14G	42,68	10,35	3,99	62,51
4N 4G	48,28	18,29	4,77	73,80
4N 9G	41,83	11,50	4,10	58,51
4N 14G	39,34	10,35	3,74	55,46
6N 4G	43,66	14,62	4,46	64,84
6N 9G	39,80	10,87	3,99	55,38
6N 14G	38,44	9,52	3,65	52,62
Menor	38,44	9,52	3,65	52,62

A etapa ***BM*** apresentou os maiores valores de tempos para todas configurações de máquina virtual quando comparado com as etapas ***KA*** e ***OS***, tendo como menor tempo o valor de 38,44 minutos. Ao observar mais atentamente, é possível notar que independentemente das configurações dos números de núcleos e memória, a ordem de relevância em função da tomada de tempo obedece sempre a mesma ordem, ou seja, na sequência ***BM***, ***KA*** e ***OS***. Para a representatividade de uma instalação completa, realiza-se a computação de todas as etapas para cada configuração de máquinas virtuais, tendo como o valor médio 62,51 minutos e como menor valor o tempo de 52,62 minutos.

De modo a observar o quão relevante se faz a capacidade computacional diante das tarefas propostas, observa-se que na Figura 5.3a, para o cenário do passo ***BM*** a quantidade de núcleos e memória interferem positivamente na tomada de tempo. A execução combinando 6 núcleos de processamento, com 14 e 9 gigabytes de memória RAM, aferiram desempenhos similares. Entretanto, com 14 gigabytes há uma distribuição de tempo mais estável, podendo ser observado pela menor distância entre o sétimo e o nono quartil. Desempenho semelhante ocorre para configuração com 4 núcleos e com 14 e 9 gigabytes de memória.

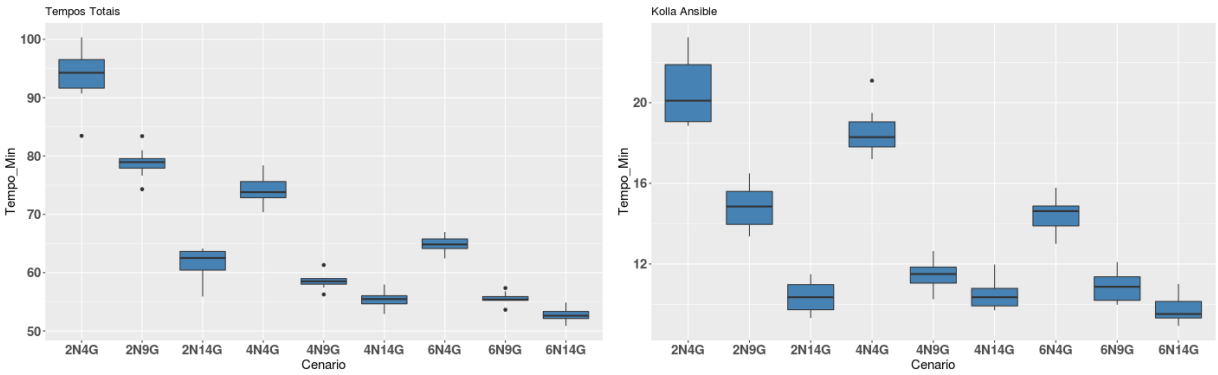
As tomadas de tempo para o passo ***KA***, observadas na Figura 5.3b, assemelham-se aos cômputos do passo ***OS***, exceto pela escala de tempo na qual a ordem de tempo do ***OS*** é menor. A similaridade do comportamento repete-se inclusive entre os *boxplots* das configurações de 6 núcleos combinados com 14 e 9 gigabytes, sinalizando homogeneidade das tomadas de tempo.

A Figura 5.3a apresenta a somatória dos resultados das etapas ***BM***, ***KA*** e ***OS***, onde demonstram a representatividade total da execução durante os testes, permitindo visualizar alguns pontos flutuantes.

Dos experimentos realizados com ***OS***, observa-se comportamento normal, o que pode ser visualizado na Figura 5.3c. Ao observar as tomadas por núcleos, analisamos que os grupos com 2, 4 e 6 núcleos, são decrescentes e para cada grupo há também uma variação decrescente conforme aumenta a quantidade de memória. Entretanto, ao se observar o comportamento entre os experimentos com 14 gigabytes de memória combinados com 6 e 4 núcleos, nota-se resultados similares em termos de estabilidade, ou seja, o conjunto de tomada de tempos possui igualdade na distribuição normal, mesmo advindo da configuração com menor número de núcleos, sugerindo que neste caso, a quantidade de memória é preponderante.

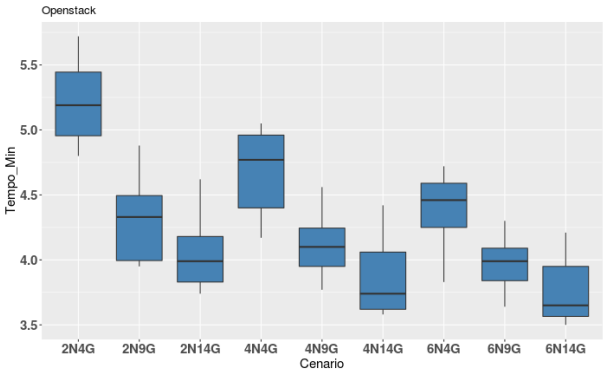
Considerando as possíveis opções de instalações: Instanciação Completa com Imagem pré-carregada (Subseção 3.3.2) e Instanciação Completa com *containers* (Subseção 3.3.3), averiguou-se os tempos para exportação e importação de cada *snapshot*, conforme a Figura 5.4. As Figuras 5.4a e 5.4b representam, respectivamente a exportação e importação do *snapshot* de uma máquina virtual nas quais já foram realizadas as configurações das suas interfaces, instalação e configuração de serviços de provisionamento *bare metal*; enquanto as Figuras 5.4c e 5.4d reproduzem, respectivamente, a exportação e importação do *snapshot* de uma máquina virtual onde já foram realizadas a instalação e configuração do Openstack Ironic (*bare metal*) e do Openstack Kolla-ansible (Controlador Openstack).

Os tempos apresentados pelas Figuras 5.4c e 5.4d, foram maiores que os das Figuras 5.4a e 5.4b já que os *snapshots* utilizados possuem maior tamanho correspondente a quantidade de mais instalações realizadas no mesmo. Apesar deste fato, pode-se perceber que durante a



(a) Bare Metal (**BM**)

(b) Instanciação com Imagem pré-carregada (**KA**)



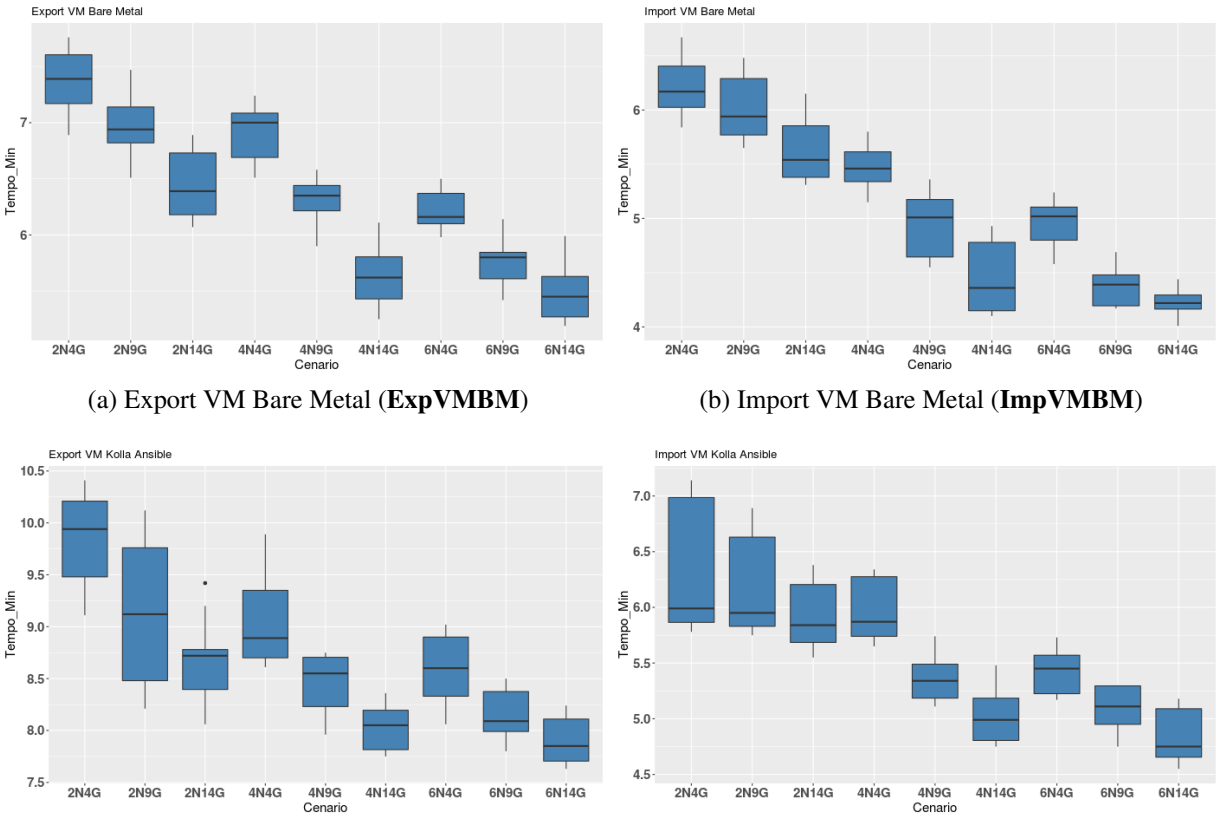
(c) Instanciação Completa com *containeres* (**OS**)

Figura 5.3: Execuções das etapas por configurações de máquinas virtuais.

realização das exportações para ambas situações (*bare metal* e kolla-ansible), suas realizações foram similares, assim como as execuções das importações. O principal fator de diferencial entre as exportações e importações foram os altos tempos de processamento apresentados durante as importações dos *snapshots* com o cenário de 2 núcleos. Quando comparado com todos os testes realizados em etapas anteriores, mesmo a situação com a menor quantidade de núcleos (2) e com a maior quantidade de memórias (14GB), acabou sendo tão eficientes quanto a uma situação de quantidade de 4 núcleos e com 4GB de memória.

Corroborando para a constatação da prevalência da quantidade de memória em detrimento do número de núcleos, notamos nas Figuras 5.5a e 5.5b, uma ligeira superioridade quanto a estabilidade, ou seja, as barras relativas às configurações com 14 gigabytes de memória sobre as configurações com 6 núcleos.

Corroborando para aconstatação da observância da prevalência da quantidade de memória em detrimento do número de núcleos, pode ser notada nas Figuras 5.5a e 5.5b, onde pode se notar uma ligeira superioridade quanto a estabilidade, ou seja, as barras relativos às configurações com 14 gigabytes de memória sobre as configurações com 6 núcleos.



(a) Export VM Bare Metal (**ExpVMBM**)

(b) Import VM Bare Metal (**ImpVMBM**)

(c) Export VM Kolla-Ansible (**ExpVMKA**)

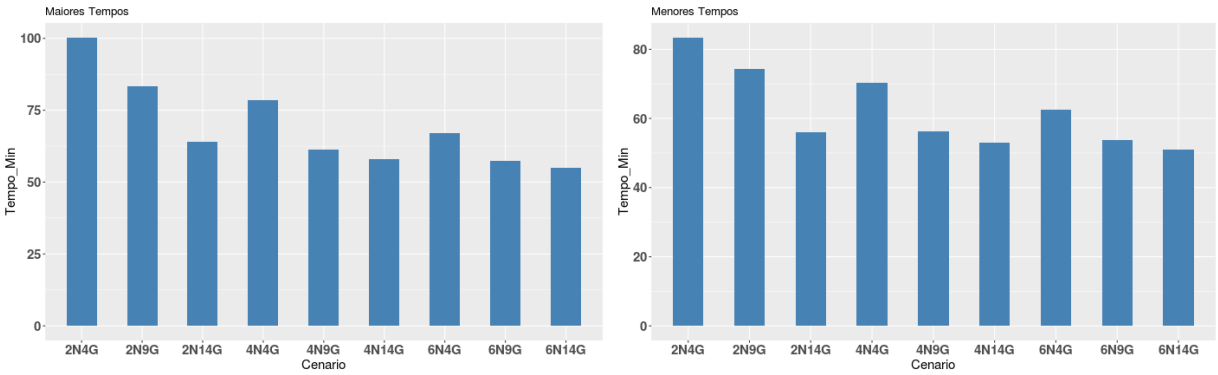
(d) Import VM Kolla-Ansible (**ImpVMKA**)

Figura 5.4: Exportação e importação de máquinas virtuais.

A Figura 5.5a apresenta os maiores tempos para cada configuração de máquina virtual durante o processo de execução dos testes ***BM***, ***KA*** e ***OS***, enquanto a Figura 5.5b, ao contrário da anterior, apresenta os menores tempos durante o processo de execução dos testes. Ambas apresentam a variação de período durante a execução para cada cenário, permitindo visualizar, embora timidamente, uma relevante diferença entre cada cenário de configurações das máquinas virtuais.

Considerando o comportamento homogêneo para os cenários apresentados até o momento é possível afirmar à questão *(i)* através dos recursos computacionais que são preponderantes na eficiência do processo e dadas nossas observações a memória principal, prevalece sobre a quantidade de núcleos.

Com relação às médias das tomadas de tempo para os experimentos, a Figura 5.3a exibe as médias e respectivas marcações de desvio padrão. É possível observar que obstante a onerosidade do tempo de instalação do *bare metal* (Figura 5.3a), os tempos dos passos de Exportação e Importação de VMs (Figura 5.4) são extremamente vantajosos em relação aos tempos de instalação do *Kolla-Ansible* (Figura 5.3b) e *Openstack* (Figura 5.3c).



(a) Maiores Tempos

(b) Menores Tempos

Figura 5.5: Tempo MAX e MIN

É notória a diferença estatística das atividades de não instalação, ou seja, importação de configurações de VMs sobre os tempos de instalação, pois os tempos das mesmas são inferiores ao tempo médio de ambas e subtraídas do desvio padrão. Deste modo, pode-se afirmar que a resposta para a questão *(ii)* aponta no sentido de que a abordagem VIM *on-demad* é fortemente vantajosa, implicando na eficiência de utilização de recursos computacionais sob esta abordagem.

Capítulo 6

CONSIDERAÇÕES FINAIS

6.1 Conclusões

Este trabalho apresentou um modelo de VIM *on-demand*, a qual possibilita instanciar máquinas a nível *bare metal* e serviços de computação, proporcionando, ao final, o seu controle e gerenciamento. Além disso, também foi possivel dividir o processo de instalação, configuração e instanciação em 3 etapas e exportar um *snapshot* para cada ambiente: (*I*) Instanciação Completa *Bare Metal*, (*II*) Instanciação Completa com *Containers* e (*III*) Instanciação Completa com Imagem Pré-Carregada.

Ao concluir os experimentos, observa-se que os melhores resultados durante os experimentos foram os que usaram memória RAM com 14 GB (G14), a maior quantidade de memória utilizada durante os testes. Diante disso, conclui-se que a quantidade de memória é aspecto preponderante no VIM *on-Demand*, pois os valores que apresentaram melhor desempenho foram em situações de maior quantidade de memória, e não alteração de outros fatores como quantidade de núcleos. Adicionalmente à preponderância da quantidade de memória, foi possível constatar que a abordagem VIM *on-demand* confere ganho computacional no tocante a eficiência de uso dos recursos.

6.2 Trabalhos Futuros

Como trabalhos futuros, pretende-se incluir outros controladores, como o VLSP e Kubernets, no VIM *on-Demand*, e realizar novos experimentos analisando consumo de energia, desempenho de memória e processamento com casos de usos que necessitem de uma sobrecarga. Comparar VIMs tradicionais com VIM *on-demand* apresentando as principais diferenças em

relaçaõ de desempenho e o que pode ser melhorado no VIM *on-demand* comparado com o tradicional. Investigar problemas de alocação de recursos que possam surgir a transferência dos *snapshots* para ser utilizado em outros servidores.

Realizar estes novos experimentos, além de interessante no ponto de vista de verificação de falhas e desempenho, permite aperfeiçoar o desenvolvimento e evolução do VIM *on-demand*, uma vez que quanto mais testes e cenários diferentes, melhores validações podem ser dadas.

6.3 Agradecimentos

O presente trabalho foi parcialmente apoiado pelo H2020, 4º Convite Colaborativo UE-BR, sob o contrato de concessão no. 777067 (NECOS - *Novel Enablers para Cloud Slicing*), financiado pela Comissão Europeia e pelo Ministério da Ciência, Tecnologia, Inovação e Comunicação (MCTIC) através da RNP e da CTIC.

O trabalho conta também com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

REFERÊNCIAS

CHIOSI, M. et al. Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action. In: SN. *SDN and OpenFlow World Congress*. [S.l.], 2012. v. 48.

CLAYMAN, S. *Network Slicing Supported by Dynamic VIM Instantatiation*. 2017.

CLAYMAN, S.; TUSA, F.; GALIS, A. Extending slices into data centers: the vim on-demand model. In: IEEE. *2018 9th International Conference on the Network of the Future (NOF)*. [S.l.], 2018. p. 31–38.

DRÄXLER, S. et al. Sonata: Service programming and orchestration for virtualized software networks. In: IEEE. *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*. [S.l.], 2017. p. 973–978.

FIFIELD, T. et al. *OpenStack Operations Guide: Set up and manage your openstack cloud*. [S.l.]: "O'Reilly Media, Inc.", 2014.

FREITAS, L. A. et al. Slicing and allocation of transformable resources for the deployment of multiple virtualized infrastructure managers (vims). In: IEEE. *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. [S.l.], 2018. p. 424–432.

GREENBERG, A. et al. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM computer communication review*, ACM, v. 39, n. 1, p. 68–73, 2008.

HAN, B. et al. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, IEEE, v. 53, n. 2, p. 90–97, 2015.

KAYOBE. *Openstack Kayobe*. 2018. Disponível em: <https://kayobe.readthedocs.io>. Acesso em: Outubro de 2018.

KIM, J.; KIM, D.; CHOI, S. 3gpp sa2 architecture and functions for 5g mobile communication system. *ICT Express*, Elsevier, v. 3, n. 1, p. 1–8, 2017.

LEMOINE, B. Itu-t network model extension for virtualized network architectures. In: IEEE. *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. [S.l.], 2018. p. 1–5.

MADEMANN, F. System architecture milestone of 5g phase 1 is achieved. *Online verfügbar unter: http://www.3gpp.org/NEWS-EVENTS/3GPP-NEWS/1930-SYS_ARCHITECTURE, letzter Zugriffam*, v. 1, p. 2018, 2017.

NECOS, P. Necos system architecture and platform specification. *MAPS- Management, Pricing and Services in Next Generation Networks: https://www.maps.upc.edu/public/NECOS 20D3.1 20final.pdf*, v. 2, p. 2018, 2018.

ORDONEZ-LUCENA, J. et al. Network slicing for 5g with sdn/nfv: concepts, architectures and challenges. *arXiv preprint arXiv:1703.04676*, 2017.

ROST, P. et al. Network slicing to enable scalability and flexibility in 5g mobile networks. *IEEE Communications magazine*, IEEE, v. 55, n. 5, p. 72–79, 2017.

SEFRAOUI, O.; AISSAOUI, M.; ELEULDJ, M. Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, Foundation of Computer Science, 244 5 th Avenue,# 1526, New York, NY 10001 ..., v. 55, n. 3, p. 38–42, 2012.

SILVA, F. S. D. et al. Necos project: Towards lightweight slicing of cloud federated infrastructures. In: IEEE. *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. [S.l.], 2018. p. 406–414.

SOARES, J. et al. Cloud4nfv: A platform for virtual network functions. In: IEEE. *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. [S.l.], 2014. p. 288–293.

TAURION, C. *Cloud computing-computação em nuvem*. [S.l.]: Brasport, 2009.

TRIPLEO. *Openstack TripleO*. 2018. Disponível em: <https://docs.openstack.org/tripleo-docs>. Acesso em: Outubro de 2018.

ZHANG, H. et al. Network slicing based 5g and future mobile networks: mobility, resource management, and challenges. *IEEE Communications Magazine*, IEEE, v. 55, n. 8, p. 138–145, 2017.

ZHOU, X. et al. Network slicing as a service: enabling enterprises' own software-defined cellular networks. *IEEE Communications Magazine*, IEEE, v. 54, n. 7, p. 146–153, 2016.

GLOSSÁRIO

API – *Application Programming Interface*

AWS – *Amazon Web Services*

BMC – *baseboard management controller*

HOT – *Orchestration Templates*

IC – *Infrastructure SDN controller*

IMA – *Abstract Infrastructure Manage*

IPMI – *Intelligent Platform Management Interface*

IaaS – *Infrastructure as a Service*

KVM – *Kernel based Virtual Machine*

NECOS – *Novel Enablers for Cloud Slicing*

NFV – *Network Functions Virtualization*

NF – *Networking Function*

NV – *Network virtualization*

PIP – *Python Package Index*

PXE – *Preboot eXecution Environment*

QoS – *Quality of Service*

REST – *Representational State*

SDN – *Software Defined Networking*

SSH – *Secure Shell*

SaaS – *Slice as a Service*

TI – *Tecnologia da Informacão*

UML – *Unified Modeling Language*

VIM – *Virtualized Infrastructure Manager*

VM – *Virtal Machine*

VNF – *Virtualized Network Function*

XML – *Extensible Markup Language*

Apendice A

INSTALAÇÃO E CONFIGURAÇÃO

Neste apêndice é apresentado o código implementado nas máquinas virtuais do controlador e *computer*, utilizando o provisionador do *Vagrant Shell* configurado no arquivo Vagrantfile que, por sua vez, permite carregar e executar os *script* de instalação, configuração e instanciação. Esse processo é responsável por instalar o Openstack Ironic, Kolla-Ansible, provisionamento da máquina Bare Metal e instalação do Openstack Tacker.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

ENV["LC_ALL"] = "en_US.UTF-8"
Vagrant.configure("2") do |config|
   config.vm.define "controller" do |controller|
      controller.vm.box = "ubuntu/xenial64"
      controller.vm.hostname = 'controller'
      controller.vm.network "private_network", ip: "10.0.0.11"
      controller.vm.network "public_network"

      controller.vm.provider "virtualbox" do |vb|
      vb.memory = "10240"
      vb.cpus = "4"
      end

      controller.vm.provision "shell", inline: <<-SHELL
         sudo su
         git clone https://github.com/henriqelol/Openstack.git --quiet
         if [ $? -ne 0 ]; then echo "Erro de Download"; fi
         chown -R vagrant.vagrant necos
```

```
        bash /Openstack/Ironic/All/update.sh
        cp /Openstack/Ironic/All/hosts /etc/
        cp /Openstack/Ironic/All/admin-openrc /etc/
        source /etc/admin-openrc
        cp /Openstack/Ironic/All/demo-openrc /etc/
        source /etc/demo-openrc
        bash /Openstack/Ironic/Controller/chrony.sh
        bash /Openstack/Ironic/Controller/mysql_py.sh
        bash /Openstack/Ironic/Controller/rabbitmq.sh
        bash /Openstack/Ironic/Controller/memcached.sh
        bash /Openstack/Ironic/Controller/etcd.sh
        bash /Openstack/Ironic/Controller/keystone.sh
        bash /Openstack/Ironic/Controller/glance.sh
        bash /Openstack/Ironic/Controller/nova.sh
        bash /Openstack/Ironic/Controller/neutron.sh
        bash /Openstack/Ironic/Ironic/All/mv-log.sh
    SHELL
  end
end
```

Apêndice A.1: Arquivo Vagrantfile Controller.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

ENV["LC_ALL"] = "en_US.UTF-8"
Vagrant.configure("2") do |config|

    config.vm.define "compute" do |compute|
        compute.vm.box = "ubuntu/xenial64"
        compute.vm.hostname = 'compute'
        compute.vm.network "private_network", ip: "10.0.0.31"
        compute.vm.network "public_network"

        compute.vm.provider "virtualbox" do |vb|
            vb.memory = "10240"
            vb.cpus = "4"
        end

        compute.vm.provision "shell", inline: <<-SHELL
            sudo su
            git clone https://github.com/henriqelol/Openstack.git --quiet
            if [ $? -ne 0 ]; then echo "Erro de Download"; fi
            chown -R vagrant.vagrant necos
            bash /Openstack/Ironic/All/update.sh
            cp /Openstack/Ironic/All/hosts /etc/
            cp /Openstack/Ironic/All/admin-openrc /etc/
            source /etc/admin-openrc
            cp /Openstack/Ironic/All/demo-openrc /etc/
            source /etc/demo-openrc
            bash /Openstack/Ironic/Compute/chrony_others.sh
            bash /Openstack/Ironic/Compute/nova_compute.sh
            bash /Openstack/Ironic/Compute/neutron_compute.sh
            bash /Openstack/Ironic/Ironic/All/mv-log.sh
        SHELL
    end
end
```

Apêndice A.2: Arquivo Vagrantfile Compute.

```
vagrant ssh controller
   bash /Openstack/Ironic/chossedriver.sh
   bash /Openstack/Ironic/config_version_api.sh
   bash /Openstack/Ironic/create_baremetal/no.sh
   bash /Openstack/Ironic/create_baremetal/valide_no.sh
   bash /Openstack/Ironic/export_implement.sh
   bash /Openstack/Ironic/name_logic.sh
   bash /Openstack/Ironic/interface_hardware.sh
```

Apêndice A.3: Instanciação *slice bare metal*.

```
vagrant ssh controller
   bash /Openstack/kolla-ansible/install_dependencies.sh
   bash /Openstack/kolla-ansible/install_Kolla_ansible.sh
   bash /Openstack/kolla-ansible/Prepare_Initial_Configuration/inventory.sh
   bash /Openstack/kolla-ansible/Prepare_Initial_Configuration/
      kolla_passwords.sh
   bash /Openstack/kolla-ansible/Prepare_Initial_Configuration/
      kolla_globals.sh
   bash /Openstack/kolla-ansible/usingOpenstack.sh
```

Apêndice A.4: Instalação Kolla-Ansible.

```
vagrant ssh controller
   bash /Openstack/tacker/install_Tacker.sh
   bash /Openstack/tacker/vim_show.sh
   bash /Openstack/tacker/vim_list.sh
   bash /Openstack/tacker/vim_update.sh
   bash /Openstack/tacker/vim_delete.sh
```

Apêndice A.5: Instalação Openstack Tacker.