CENTRO UNIVERSITÁRIO DE BRASÍLIA - UNICEUB
FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS - FATECS
CURSO DE ENGENHARIA DE COMPUTAÇÃO

ÉVERTON VERÍSSIMO ALVES DE MELO

AUTOMAÇÃO E PROVISIONAMENTO DE UMA REDE CAMPUS UTILIZANDO OPEN NETWORKING

Orientador:
Prof.º. MSc. Francisco Javier De Obaldía Diaz

Brasília
JUNHO/2018

ÉVERTON VERÍSSIMO ALVES DE MELO

AUTOMAÇÃO E PROVISIONAMENTO DE UMA REDE CAMPUS UTILIZANDO OPEN NETWORKING

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Orientador: Prof. MSc. Francisco Javier De Obaldía Diaz

Brasília

JUNHO/2018

ÉVERTON VERÍSSIMO ALVES DE MELO

AUTOMAÇÃO E PROVISIONAMENTO DE UMA REDE CAMPUS UTILIZANDO OPEN NETWORKING

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientador: Prof. MSc. Francisco Javier De Obaldía Diaz

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas-FATECS.

_____

Prof. Abiezer Amarilia Fernandes
Coordenador do Curso

Banca Examinadora:

_____

Prof. MSc. Francisco Javier De Obaldía Diaz
Orientador

_____

Prof. nome, titulação.
Instituição

_____

Prof. nome, titulação.
Instituição

_____

Prof. nome, titulação.
Instituição

AGRADECIMENTOS

Gostaria de agradecer as pessoas que fizeram parte desta jornada. Meus pais e minha irmã, que contribuíram e me incentivaram para conclusão deste curso. A Sinara pelo seu apoio e carinho que fizeram a realização deste trabalho possível.

Agradeço também ao professor Francisco Javier pela sua orientação e tempo dedicado a este projeto.

RESUMO

O aumento da complexidade e tamanho dos ambientes de centro de dados gerou a necessidade de novos modelos cada vez mais eficientes, flexíveis e escaláveis. O Facebook pensando em superar esses desafios, iniciou uma comunidade colaborativa denominada Open Compute Project – OCP. O resultado desta iniciativa levou a um redesenho de todos os componentes de um centro de dados. Para os componentes de rede, surgiu a solução que desacopla o software do hardware, de provisionamento automático e automação nos processos de gerenciamento e configuração. Essa nova abordagem altera a arquitetura dos modelos tradicionais e proprietários. Buscando uma arquitetura aberta tanto para hardware quanto para software, permitindo assim, que os equipamentos de rede passem a ser gerenciado da mesma forma que os servidores, utilizando ferramentas de provisionamento, orquestração e gerência de configuração como Puppet, Saltstack e Ansible. As empresas podem desfrutar de benefícios como redução no tempo de implantação, redução de custos e melhorando performance dos seus ativos. Assim, este trabalho propõe um estudo sobre a arquitetura Open networking, ferramentas de automação e provisionamento, buscando implementar estes conceitos em uma rede Campus. Como resultado final espera-se implementar um software de automação integrado com um sistema operacional de rede baseado em Linux instalado em um emulador, com o objetivo de reduzir o tempo de implantação e manutenção deste tipo de solução.

Palavras Chave: OPEN NETWORKING, AUTOMACAO, PROVISIONAMENTO , NETDEVOPS.

ABSTRACT

The increasing the complexity and size of datacenter environments has generated the need for new, more efficient, flexible and scalable models. Facebook thinking about overcoming these challenges, started a collaborative community called Open Compute Project - OCP. The result of this initiative has led to a redesign of all the components of a datacenter. For the network components, a solution has emerged that disaggregated the hardware & software, automatic provisioning and automation in the management and configuration processes. This new approach changed the architecture of both traditional and proprietary models. Seeking an open architecture for both hardware and software, allowing network equipment to be managed in the same way as servers, using provisioning tools, orchestration and configuration management such as Puppet, Saltstack and Ansible. Companies can enjoy benefits such as reduced deployment time, reduced costs and improved performance of their assets. Therefore, this work proposes a study on the Open networking architecture, automation and provisioning tools, seeking to implement these concepts in a Campus network. As a final result, it is expected to implement integrated automation software with a Network Operacional System Linux-based installed in a emulator, in order to reduce the time of deployment and maintenance of this type of solution.

Keywords: OPEN NETWORKING, AUTOMATION, PROVISIONING, NETDEVOPS.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLA

ASIC - Application-Specific Integrated Circuit

API - Application Programming Interface

BGP - Border Gateway Protocol

CAN - Campus Area Network

CLI - Command Line Interface

CPU - Central Processing Unit

DARPA - Defense Advanced Research Projects Agency

DHCP - Dynamic Host Configuration Protocol

ECMP - Equal-Cost Multi-Path Routing

FIB - Forwarding Information Base

GNS3 - Graphical Network Simulator-3

NCLU - Cumulus Linux Network Command Line Utility

NDA - Non-Disclosure Agreement

NOS - Network Operational System

SDN - Software-Defined Networking

OCP - Open Computer Project

ODM - Original Design Manufacturers

ONIE - Open Network Install Environment

ONVL - Open Netvisor Linux

QoS - Quality of services

RIB - Routing Information Base

SDRAM - Synchronous Dynamic RAM

STP - Spanning Tree Protocol

U-boot - Universal Boot Loader

VLAN - Virtual Local Area Network

VM - Virtual Machine ou Máquina Virtual;

YAML - Ain't Markup Language

ZTP - Zero touch provisioning

SUMÁRIO

CAPÍTULO 1 – INTRODUÇÃO

Neste capítulo serão apresentados uma breve explanação sobre o tema, juntos com as motivações, objetivos, e justificativas que levaram a construção deste trabalho, também serão apresentados o escopo do trabalho e as metodologias utilizadas no desenvolvimento do trabalho. Por fim, serão descritos a estrutura do trabalho.

1.1. APRESENTAÇÃO DO PROBLEMA

O surgimento da primeira rede de computador baseada em computação de pacotes deu-se através de uma requisição do departamento de defesa americano à DARPA (Defense Advanced Research Projects Agency). Eles buscavam uma rede que fosse capaz de sobreviver a uma guerra nuclear. A primeira demostração desta rede foi realizada em dezembro de 1969, quando cientistas de 4 universidades americanas (University of California, Los Angeles - UCLA, University of California, Santa Barbara – UCSB, Stanford Research Institute – SRI e University of Utah) conseguiram transmitir e receber informações. A ARPANET com ficou conhecida esta rede de pesquisa evoluiu se conectando a novas redes e de acordo com as necessidades novos protocolos, como o TCP/IP, foram criados. (TANENBAUM, 2003)

Essa rede foi a base para o que hoje conhecemos como Internet, e desde então poucas evoluções foram observadas, o autor Miller enfatiza

Segundo Miller:

> "Arquiteturas e infraestruturas tradicionais de rede permanecem estáticas há mais de 30 anos. Em nenhum outro lugar no mundo da tecnologia da informação é provável que você encontre tal estagnação [...]. Os elementos de computação e os elementos de armazenamento avançaram rapidamente ao longo dos anos, aproveitando as tecnologias de virtualização e abstração para reduzir sua carga e aumentar seu valor [...]. Hoje, a natureza rígida das redes legadas é um gargalo nos negócios (alguns podem até dizer "O" gargalo dos negócios), sufocando a agilidade e inovação dos negócios, enquanto frustra as equipes de rede que devem manter essas arquiteturas monolíticas." (MILLER, 2016, p.1, tradução nossa)

1.2. MOTIVAÇÃO

Em 2009, o *Facebook* preocupado com a evolução do seu ambiente computacional começou a repensar todos os componentes do seu próximo *centro de dados*. Como resultado deste esforço de dois anos foi construído o centro de dados de Prineville, no Oregon. Ele foi 24% mais barato e 38% mais eficiente na utilização de energia. Em 2011, o Facebook compartilhou seus projetos com o público e, junto com a Intel, Rackspace, Goldman Sachs e Andy Bechtolsheim, lançou o Open Compute Project e incorporou o Open Compute Project Foundation. O objetivo era cria um movimento open para os componentes de hardware, assim como o que ocorreu com o software livre, com o Linux, onde outras empresas também pudessem contribuir. (OCP,2017)

A responsável por este crescimento foi a necessidade de atender as novas demandas do seu modelo de negócios, a quantidade de informação que deveria ser processada e armazenada passou a ser cada vez maior. Novas técnicas tiveram que ser aplicadas com forma de melhorar a utilização dos seus recursos.

A virtualização baseada em *hypervisor* inicialmente introduzida pela VMware em 2001, trouxe um nível de abstração da camada de hardware permitindo que os servidores passassem a ser gerenciados com uma flexibilidade nunca vista antes. (MARSHALL; REYNOLDS; MCCRORY, 2006). Porém essa evolução não se refletiu na arquitetura de redes que basicamente pouco evoluiu ao longo dos últimos trinta anos. (MILLER, L.C. 2016)

O modelo tradicional de rede composto por uma arquitetura fechada de hardware e software se tornou um dos principais gargalos para a expansão dos novos centros de dados. Isso porque além do alto custo dos equipamentos, a difícil configuração, difícil escalabilidade e o aumento na quantidade de equipamentos que passou de centenas para milhares, tornou quase impossível o gerenciamento individual destes ativos.

Após a criação do Open Networking Foundation e Open Compute Project surgiram algumas ideias que sugeriram melhorias na utilização dos recursos de rede. Entre elas tem-se a separação do software em relação ao hardware nos equipamentos de rede; algo parecido com o que já tinha acontecidos com os servidores baseado na arquitetura x86, onde era possível a escolha do sistema operacional. Outro ponto de melhoria, foi o provisionamento automático e automação nos processos de gerenciamento e configuração.

Essa última ideia relacionada a automação foi derivada do conceito de DevOps, em inglês, que tem o objetivo de unir a palavra desenvolvimento (Dev) com a palavra operações (Ops). Um DevOps normalmente é conhecido por utilizar softwares de automação de infraestrutura,

que pode ir de serviços como configuração de um DNS até o gerenciamento de redes (GREENE, 2015). O objetivo principal deste movimento é acabar com os desentendimentos entre áreas de desenvolvimento e de operações, fazendo com que a criação de serviços fosse mais ágil e livre de erros (WILSON, 2015). Devido ao nascimento do movimento DevOps, frameworks de automação como Puppet, Chef e Ansible surgiram para ajudar na automação da infraestrutura (LERNER, 2014).

Baseados em todos estes conceitos, este trabalho teve a motivação de reunir estes conhecimentos que já são implantados em larga escala nos grandes centros de dados e aplica-los em uma rede campus que apesar de ter um menor tamanho também tem a necessidade de redução de custos, fácil provisionamento e simples configuração.

1.3. OBJETIVOS DO TRABALHO

1.3.1. OBJETIVO GERAL

Implementar uma solução que mostre a redução da complexidade de instalação de switches em uma rede Campus Area Networking (CAN) utilizando um software automação.

1.3.2. OBJETIVO ESPECÍFICOS

- Implementar um software de automação para reduzir o tempo de implantação e manutenção;
- Desenvolver o ambiente de teste para agilizar a configuração e provisionamento da rede;
- Avaliar a redução de tempo na implantação e gerenciamento.

1.4. JUSTIFICATIVAS E IMPORTÂNCIA DO TRABALHO

As justificativas para o desenvolvimento do trabalho são baseadas nos conhecimentos adquiridos ao longo desta pesquisa. É sabido que grande parte das evoluções tecnológicas que ocorreram nos últimos anos advêm das limitações encontradas nos grandes centros de dados; soluções foram propostas e tem contribuído significativamente para forma de gerenciamento e provisionamento desses ativos. Entretanto, essas evoluções não têm refletido nas

implementações de redes campus. Em geral, as atualizações são feitas através de um único fabricante com hardware e software proprietários, o que gera uma dependência, alto custo, pouco gerenciamento e quase nenhuma automação.

Assim este trabalho contribui, para uma mudança neste cenário, tratando de ferramentas abertas que podem ser utilizadas para automação e provisionamento de equipamentos de rede. Temas como open networking e Software Defined Network (SDN) são abordados para que se possa ter o entendimento das tecnologias, atualmente possui uma infinidade de projetos que propõem mudanças na forma de gerenciamento, o tema open networking muitas vezes é explorado de uma forma equivocada não possibilitando a sua separação do tema SDN. Este trabalho demonstra que é possível a utilização de um equipamento open networking sem a utilização de SDN. Por fim, importante mencionar que este trabalho pode servir com material de pesquisa para estudantes e administradores de redes que querem aprofundar os conhecimentos neste assunto.

1.5. ESCOPO DO TRABALHO

O escopo do projeto consiste em implementar uma solução que permita a automação e provisionamento de equipamentos em uma rede campus. O trabalho emulará o funcionamento de uma topologia de rede permitido realizar o provisionamento, automatizar processos de configuração com criação de VLANs, configuração de roteamento, identificação de problemas ou *troubleshooting* e por último, criação de rotina de backup.

Dentro deste escopo, não se contempla a implementação em equipamentos reais, devido o alto custo envolvido, contudo observa-se que os softwares de emulação estão cada vez mais próximos do ambiente real e vem sendo amplamente usado em automações, como forma de prever problemas e planejar implementações.

Também não será o foco deste trabalho, propor modelo de implementação ou formas de configuração em um cenário real.

1.6. TRABALHOS CORRELATOS

Ao longo deste trabalho de pesquisa, foram encontradas inúmeras referências relacionadas aos conceitos SDN, engenharia de tráfego e automação.

Trabalho como a dissertação de mestrado com o tema Arquitetura e Protótipo de uma rede SDN-Openflow para provedores de serviço, feito por (RODRÍGUEZ, 2014) que busca estudar

alternativas para construção de lógicas de roteamento e encaminhamento de tráfego para redes de grande porte como operadoras.

Outro trabalho que explora um tema relacionado, e foi usado como referência, é o trabalho de conclusão de curso com tema Estudo e Implementação de soluções para automação de dispositivos de rede, feito por Daniel Cristiano Menzen (MENZEN, 2015). Neste trabalho ele busca demonstrar alguns benefícios da automação em rede utilizando equipamentos com o software de automação Ansible.

Alguns artigos científicos também foram utilizados como referência para este trabalho; dentre eles tem-se o estudo com o tema Abordagem SDN: uma mudança de paradigma na arquitetura de rede tradicional que busca apresentar um novo conceito para o tradicional. (GIESEN, OLIVEIRA, 2017)

Todos estes estudos foram importantes para a pesquisa e aprimoramento dos conceitos relacionados ao tema deste trabalho. Porém nenhum abordava os conceitos relacionados a open netwoking ou automação, buscado avaliar os aprimoramentos tecnológicos desenvolvidos para o ambiente de centro de dados em um contexto de redes campus utilizando emulação para avaliar e testar essa nova arquitetura. Outro ponto que ainda não foi muito explorado e que este trabalho busca clarificar é a diferenciação do conceito de open networking e Software-Defined Networking ou SDN.

1.7. RESULTADOS ESPERADOS

Como resultado esperado do projeto, tem-se a implementação de um software que possibilite a automação em uma rede CAN (Campus Area Network), reduzindo o tempo de implantação e manutenção. A simulação será feita com uso de um sistema operacional de rede ou NOS (Network Operational System), que possibilite validar a redução de custos relacionado ao hardware e software não proprietários. Também se espera que seja possível efetuar as configurações para o correto funcionamento do software de emulação GNS3 e sua integração com o software de virtualização VMware Fusion. Permitindo o desenvolvimento de um ambiente de teste, onde seja possível extrair as informações sobre o desempenho da implementação e tempo de implantação.

1.8. METODOLOGIA

A metodologia utilizada para a composição deste trabalho. A proposta principal desta pesquisa é propor uma solução para problemas reais existentes em redes campus tornando assim a sua natureza aplicada. Quanto aos objetivos essa pesquisa este trabalho possui uma característica exploratória, pois através do levantamento bibliográfico pode-se criar familiaridade com o fato levando a um melhor entendimento, podendo assim estabelecer relações entre problema e a arquitetura proposta. Quanto a sua abordagem este trabalho possui uma característica quantitativa, pois os resultados obtidos serão analisados buscando avaliar a redução de tempo na instalação e manutenção de ativos de rede. (OTANI, FIALHO; 2011).

A construção deste trabalho foi dividida em 3 etapas: Na primeira, referente ao planejamento foi escolhido o tema baseado em conhecimentos empíricos, passando a criar hipótese e delimitar problema; Na segunda, referente a execução, foram sugeridos os objetivos, passando a definir os procedimentos e recursos necessários. Assim foi necessário criar a fundamentação teórica, para análise e estratificação dos dados para análise e conclusão. Na terceira e última etapa, foi referente a construção do texto científico buscando uma coerência lógica que pudesse ser clara e informativa. (OTANI, FIALHO; 2011)

1.9. ESTRUTURA DO TRABALHO

A monografia é composta por cinco capítulos.

Capítulo 1 - Trata o tema abordado, fazendo uma introdução do tema. O texto ainda apresenta a justificativa para do projeto, escopo e trabalhos correlatos;

Capítulo 2 - Apresenta a contextualização do problema mediante ao cenário atual das tecnologias existentes e aponta problemas enfrentados no mercado também e explorado todo o referencial teórico necessário para a compreensão do que foi desenvolvido;

Capítulo 3 - Refere-se à arquitetura proposta, juntamente com a sua explicação e funcionamento;

Capítulo 4 - Aborda a implementação do cenário de testes para o funcionamento do projeto;

Capítulo 5 - Considerações finais: conclui o trabalho, avaliando a solução propostas e aponta sugestões para futuros trabalhos.

CAPÍTULO 2 – REFENCIAL TEÓRICO

Neste capítulo será apresentado a contextualização do problema mediante ao cenário atual das tecnologias existentes e apontar problemas enfrentados no mercado também, assim como é explorado todo o referencial teórico necessário para a compreensão do que foi desenvolvido, fazendo a diferenciação de alguns conceitos necessários para o entendimento deste trabalho.

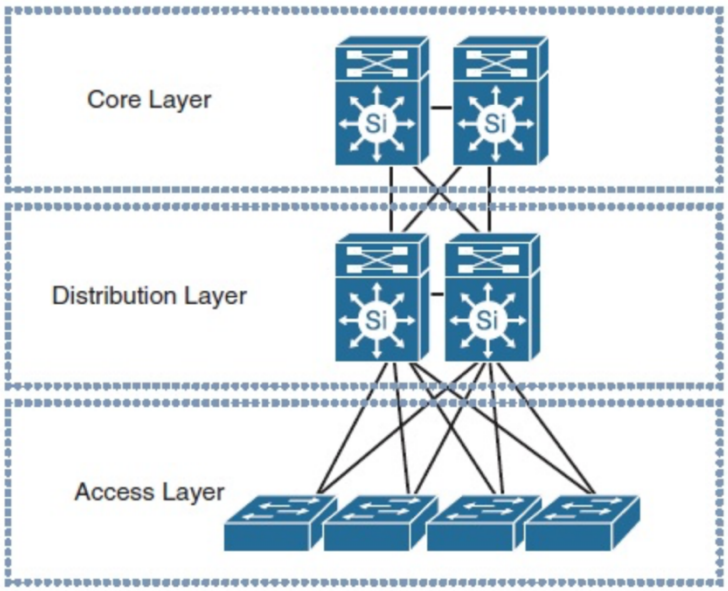
2. REDES CAMPUS

Uma rede campus é geralmente compostas por equipamentos corporativos e possui uma alta demanda de tráfego de dados e variados tipos de serviços. A rede campus pode ser composta por um ou mais edifício concentrado em uma única localidade geográfica que pode se estender por alguns quilômetros. (AL-SHAWI, 2016)

Segundo Al-shawi, quando possível três aspectos devem ser observados no projeto de uma rede campus. São eles a hierarquia, modularidade e resiliência.

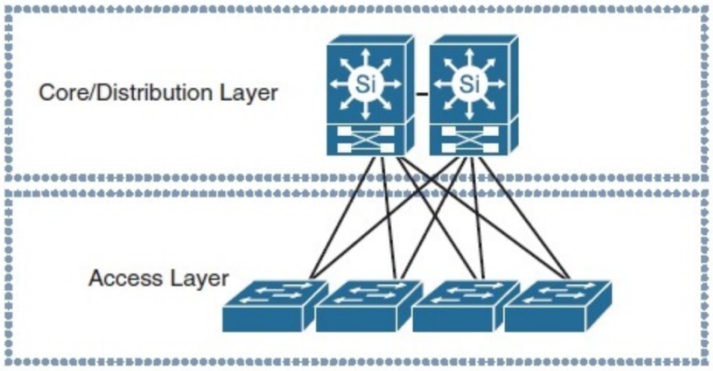
No que se trata da hierarquia temos dois modelos: a arquitetura de três camadas ou a arquitetura de duas camadas, conforme pode ser visualizado nas Figuras 1 e 2 respectivamente. A arquitetura de três camadas é composta pela camada de núcleo onde são realizadas as funções de transporte entre sites e roteamento de alto desempenho. A segunda camada é a de distribuição onde é realizada a agregação de links e a responsável pela comunicação entre a camada de acesso e núcleo. A última camada é a de acesso, onde são conectados os usuários.

Figura 1 - Modelo de hierarquia com três camadas



Fonte: Al-shawi (2016)

Figura 2 - Modelo de hierarquia com duas camadas



Fonte: Al-shawi (2016)

Ainda segundo Al-shawi, se tratando de uma pequena ou média rede pode-se optar pelo modelo de duas camadas, onde as funções de núcleo e distribuição passam a ser realizadas pela mesma camada. Esta topologia traz algumas melhorias em relação ao modelo de três camadas, com a redução da latência no tráfego Norte-Sul, caracterizado pelo tráfego que sai ou entra da rede para Internet, tráfego típico de clientes e no tráfego Leste-Oeste caracterizado pelo tráfego entre servidores, tráfego típico de um centro de dados. A retirada de uma camada diminui o tempo de processamento dos pacotes reduzindo a latência. Outra melhoria na utilização desta topologia é a eliminação do protocolo STP (Spanning Tree Protocol), atualmente existem protocolos de camada 2 como o MLAG ou protocolos de camada 3 como OSPF, ECMP ou BGP que podem utilizar múltiplo caminhos sem a criação de *loop* na rede (METZLER, 2011).

A utilização do modelo de duas camadas traz uma redução nos custos de aquisição uma vez que a camada de núcleo acumula a função da camada de distribuição. Por este motivo a topologia proposta neste trabalho se baseará na hierarquia de duas camadas, com a modularidade de se conectar a diferentes tipos de redes como Datacenter, WAN ou Internet, com resiliência entre os caminhos que conectam camada núcleo a de acesso. (AL-SHAWI, 2016)

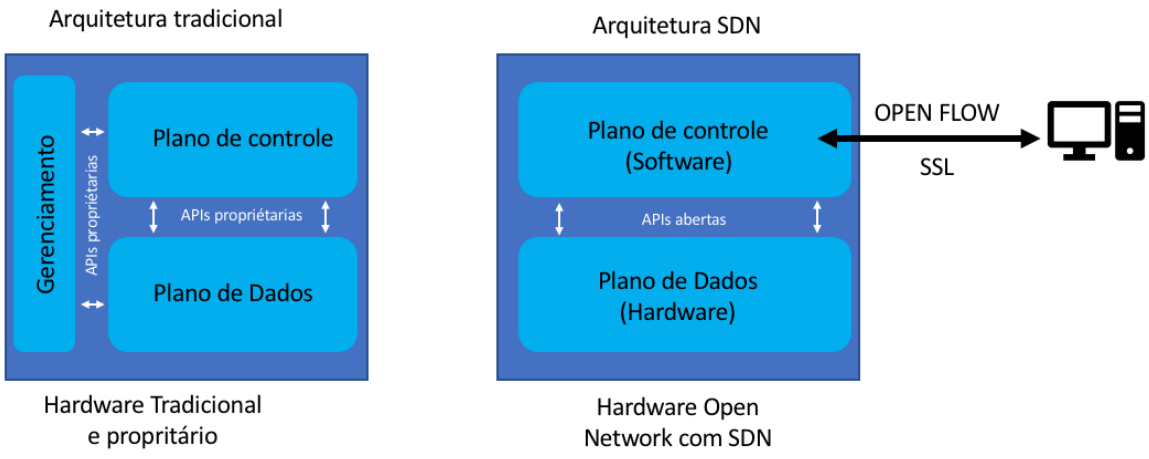
2.1. SOFTWARE DEFINED NETWORKING

Com o crescimento da demanda por novos serviços, aplicações e quantidade de dados a serem armazenados nos novos centros de dados, alguns requisitos com eficiência, escalabilidade e custos passaram a não serem mais atendidos pelo modelo tradicional de rede. Em 2008, cientistas da Universidade de Berkeley e Stanford sugeriram uma nova abordagem

para o gerenciamento de rede de computadores. Esta nova abordagem ficou conhecida com redes definidas por software ou Software Defined Networking (SDN) **(**GIESEN, OLIVEIRA, 2017).

Uma das premissas para a criação deste modelo em uma fase inicial foi a desagregação do hardware e do software dos equipamentos. Os modelos tradicionais de rede desde então utilizavam hardware e software proprietários e um único fabricante (ONF,2018). Isso levava a um modelo onde o plano de controle representado pela RIB (Routing Information Base) responsável pelas decisões de roteamento e o plano dados representado pela FIB (Forwarding Information Base) onde os dados são comutados em camada 2 trabalhavam um conjunto integrado em um mesmo equipamento. A proposta do SDN foi que os equipamentos de rede passassem a ter apenas plano de dados, melhorando a performance e diminuindo a complexidade da rede. As funções realizadas no plano de controle passariam a ser executadas por um novo elemento de rede denominado de controlador. Este elemento teria uma visão e controle total da rede possibilitando a tomada de decisões, criação de instâncias, adicionar ou remover entradas na tabela de fluxo, atribuir prioridade dinamicamente a diferentes fluxos de dados e a possibilidade de integração com diferentes APIs. A comunicação entre o controlador e o equipamento de rede seria realizado através do protocolo OPENFLOW, um protocolo aberto que poderia ser aderido por diferentes fabricantes, trazendo uma padronização para o mercado. Abaixo podemos observar a proposta do SDN na Figura 3.

Figura 3 – Comparativo entre arquitetura tradicional e SDN



Fonte: (ONF, 2014 com adaptações).

O SDN traz muitas melhorias para os ambientes de grande porte como provedores de serviços e grandes redes, onde existe a necessidade de se ter redes com várias instâncias, capaz de suportar grandes quantidades de tráfego e com tipos com diferentes tipos de QoS, VLANs e ACLs. Os provedores também possuem a necessidade de comunicação com diferentes tipos de tecnologia com LTE, PSTN, WDM (RODRÍGUEZ, 2014). Suas as redes comumente usam tabelas de roteamento (RIB) e tabelas de comutação (FIB), que possuem milhares de entradas exigindo equipamentos de grande porte que, em geral, são muito caros (SHEN, 2014).

Essas características não são validas para um ambiente de redes campus, onde comumente possui algumas centenas de equipamentos e as tabelas de roteamento e comutação com algumas centenas de entradas. Por esse motivo, a utilizando da arquitetura SDN em uma rede campus não se justifica inicialmente, embora a utilização de equipamentos open networking permita a utilização futura desta tecnologia sem a troca de hardware.

Levando em consideração o ambiente de uma rede campus, percebemos que algumas características podem ser derivadas dos ambientes de operadoras como a flexibilidade, quando da utilização de diferentes tipos de fabricantes, como a eficiência, na utilização de recursos e equipamentos, como da escalabilidade, quando a necessidade de expandir a rede, de forma rápida com o de automação de tarefas deixando mais ágeis as configurações de equipamentos e ainda reduzindo a possibilidade de erros humanos. Todos estes itens podem se alcançados com a utilização de equipamentos open networking e conjuntos com software de automação e provisionamento de rede.

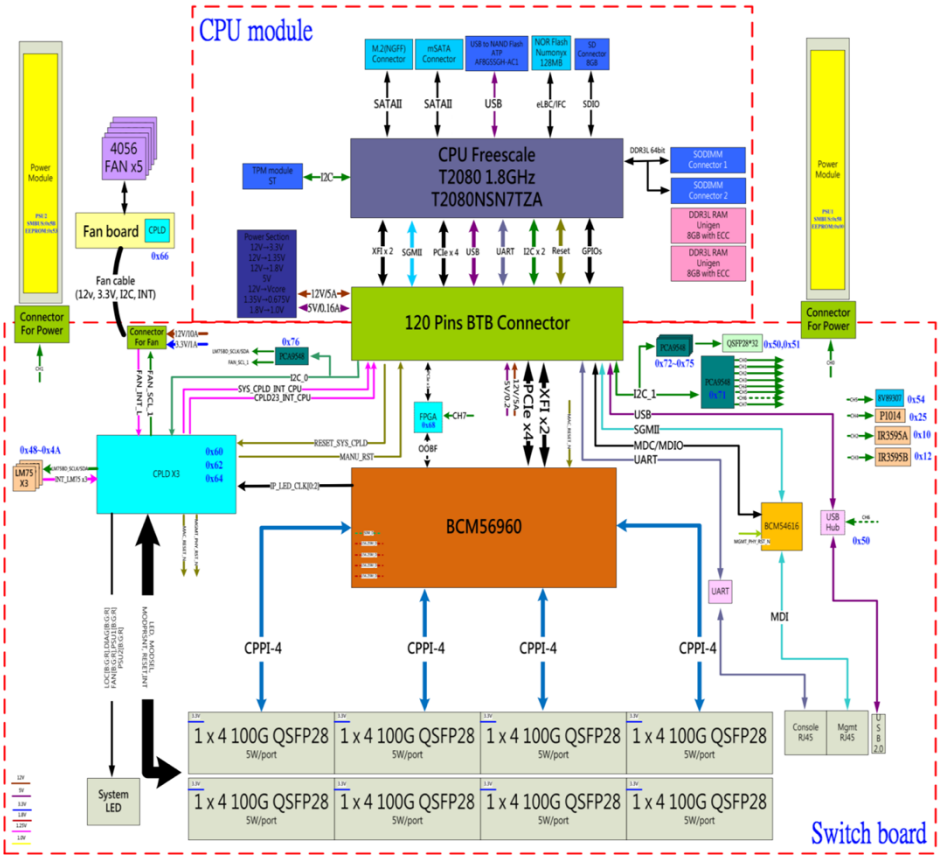
2.2. OPEN NETWORKING

De acordo com os aspectos demostrados anteriormente, tem-se que o open networking tem uma ligação com os conceitos de SDN deste o seu surgimento, buscando em uma fase inicial desagregar o hardware do software. Porém o SDN foi criado com intuito de resolver alguns problemas encontrados nos grandes ambientes como grandes rede e ambiente de operadoras. Nestes cenários são encontrados desafios no encaminhamento de pacotes, na classificação de serviços e na forma de roteamento. Os protocolos existentes não eram mais adequados para resolução destes desafios, precisando assim de novas abordagens conforme destacados anteriormente no item 2.2. Isso gera um entendimento equivocado, de que se você utilizar equipamentos open networking você deverá mudar a sua arquitetura de rede passando a utilizar o protocolo OPENFLOW com descrito na especificação do ONF (Open Networking Foundation que padroniza o SDN (ONF, 2014). Hoje, conforme este trabalho pretende

demonstrar é possível a utilização de equipamentos cujo o hardware é padronizado pelo OPC (Open Computer Project) conforme será demostrado no item 2.3.1. O OPC tem contribuído para a criação de uma comunidade colaborativa que projeta e disponibiliza os projetos de hardwares de switches para empresas que querem produzir estes equipamentos (OPC, 2018).

Todo switch tem em sua estrutura principal dois componentes: o módulo de CPU onde são processadas todas as informações referentes ao controle do hardware, é o módulo de switch, que é controlado por um hardware conhecido como ASIC (Application-Specific Integrated Circuit) que tem a função de comutar pacotes em alta velocidade (OPC, 2016). Na figura 4 abaixo, é demostrado um diagrama de blocos de um switch com 32 portas 100GB projetado pelo fabricante Accton é publicado pelo OPC. Pode-se observar os dois componentes principais sendo destacado em laranja o ASIC com o nome BCM56960 fabricado pela Broadcom, conhecido pelo codinome Tomahawk (OPC, 2016). Outro exemplo de ASIC que é utilizado em equipamentos de fabricantes como Cisco, Arista, Juniper, Huawei é o BCM56850, conhecido pelo codinome Trident 2 (TOGHRAEE, 2016).

Figura 4 - Diagrama de blocos switch



Fonte: OPC, 2016.

Atualmente a grande maioria dos fabricantes de switches do mercado utilizam ASICS de um grupo limitado de fabricantes. (TOGHRAEE, 2016). Ou seja, independe do fabricante de switch que você utilize, os ASICS serão fabricados pela Broadcom, Mellanox, Marvel, EZChip, Microsemi ou Cavium. Apenas estes fabricantes produzem estes componentes. Isso levou o Facebook, em 2009, a sugerir soluções que trouxessem novas possibilidades, propondo uma separação entre o hardware e software dos switches. Assim foi criado o Open Computer Project Foundation.

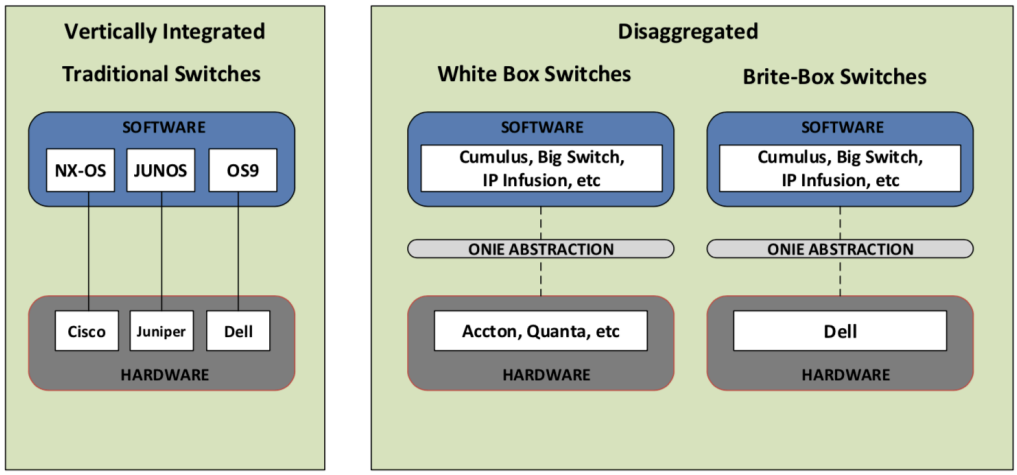
2.2.1. OPEN COMPUTE PROJECT

Nascido de um projeto liderado pelo Facebook em 2009, e disponibilizado publicamente em 2011, que buscava um novo designer para seu novo centro de dados com a maior eficiência energética e capaz de lidar com uma escala de dados nunca vista antes, foi criado o OPC. O OPC busca colaboração de ideias para construção de componentes do centro de dados cada vez mais eficientes e escaláveis. (OPC, 2018)

Para alcançar essa eficiência e escalabilidade, o OCP propôs uma alteração o atual designer das redes baseados em hardwares totalmente fechados e proprietários. Este modelo definido por cada fabricante mostra-se ineficiente, uma vez que trava tanto o rápido crescimento e a possibilidade de escolha do sistema operacional de rede. Para que seja possível montar ou crescer sua rede com equipamentos de diversos fabricantes e ainda com a possibilidade de escolher as funcionalidades que desejar com rapidez e facilidade. O OCP sugere como solução a desagregação ou separação entre os elementos de hardware e sistema operacional em um switch. (OPC, 2018)

Basicamente, hardware e software passam a ser independentes. Assim, um hardware de um determinado fabricante passa a ser capaz de receber múltiplos sistemas operacionais, e da mesma forma, um sistema operacional é suportável em vários equipamentos. A Figura 5 mostra um comparativo entre a arquitetura tradicional, em que software e hardware são proprietários de um mesmo fabricante e a mudança de arquitetura proposta pelo OCP, em que é possível escolher o software para a plataforma de hardware desejada.

Figura 5 - Modelo verticalmente integrado (tradicional) e o modelo desagregado (proposta open networking)



Fonte: LAMA, 2016.

Com a liberdade de escolha entre o hardware e software proposta pela arquitetura OCP, surge a seguinte classificação, na forma de montagem de um switch: São elas: bare metal, white box e bright box. As especificações são apresentadas a seguir.

2.2.2. BARE METAL SWITCH

Um switch com especificação bare metal não possui sistema operacional atrelado ao seu hardware, de forma que a escolha do sistema operacional dependerá das aplicações desejadas para o equipamento. O bare metal switch é fabricado por grandes fornecedores comumente chamados de *original design manufacturers* - ODM que também vendem seus equipamentos para outros fabricantes. São exemplos destes fabricantes empresas como Accton, Quanta QCT, Delta Computer e Alpha Networks. Esta arquitetura é semelhante à que se tem disponível hoje nos servidores, PC's e laptops. (DOYLE, 2015)

Outra característica importante dos switches bare metal é que eles já vêm com o boot loader, chamado ONIE – Open Network Install Environment, que permite a instalação do sistema operacional de escolha. O ONIE será detalhado no item 2.6.1.

2.2.3. WHITE BOX SWITCH

Um switch white box já vem com sistema operacional instalado de fábrica, mas integrado ao hardware, possibilitando que a estrutura permanece aberta para alterações futuras

de software. Basicamente, um switch white box é um bare metal com sistema operacional já instalado, conforme mostra a Figura 5 (DOYLE, 2015).

2.2.4. BRITE BOX SWITCH

Um switch brite box, e uma mistura da palavra *branded* com *white*, ele já vem com sistema operacional instalado, porém seu hardware é fabricado por um ODM e vendido por outros fabricantes. Exemplos destes fabricantes são a HP, DELL e EDGECORE. Eles também permitem a substituição do sistema operacional de rede, conforme mostra a Figura 5 (DOYLE, 2015).

2.3. SOFTWARES DE EMULAÇÃO

Atualmente existe uma variedade de software de simulação e emuladores de rede. Essas ferramentas têm evoluído e sua utilização extrapola os ambientes de laboratório em quem os alunos iniciantes tinha o seu primeiro contato com os equipamentos. Hoje os engenheiros de rede utilizam essas ferramentas para projetar soluções e prever problemas. É possível montar uma topologia em produção, replicando as suas configurações, e executar a análise sem efetivamente ir a campo, isso gera uma flexibilidade na montagem do ambiente e uma redução de custos, uma vez que não é necessário a aquisição de equipamento para realização de teste ou validação das funcionalidades. (SOUSA, SILVA, SOUZA, CABRAL, ROCHA, OLIMPIO, 2016)

No que tange aos softwares de simulação e emuladores, é importante diferenciar e compreender essas ferramentas. Os softwares simuladores reproduzem apenas algumas características dos equipamentos reais, ou seja, existem algumas limitações. Já os softwares emuladores tentam replicar as funções de um equipamento real tentando se aproximar ao máximo do equipamento real. (SOUSA, SILVA, SOUZA, CABRAL, ROCHA, OLIMPIO, 2016)

Devido a característica *open source* deste trabalho, optou-se por utilizar uma ferramenta que tivesse seu código aberto, que pudesse se aproximar ao máximo de um ambiente real, que suportasse diferentes plataformas. Devido a estas características a ferramenta escolhida foi o GNS3 que possui duas formas de emulação. Uma utiliza apenas o software GNS3 e a outra utiliza um ambiente virtualizando que traz melhorias em ambientes de teste complexos, pois a ferramenta de virtualização passa a gerenciar os recursos de processamento e memória. Para a

utilização de virtualização a ferramenta GNS3 se integra com diferentes softwares de virtualização. Entre eles temos o software Virtual Box e o VMware Fusion quando utilizado o sistema operacional MacOS. Outra característica que levou a escolha do conjunto VMware Fusion e GNS3, foi o suporte a nested virtualization. Essa funcionalidade permite a criação de VMs dentro de VMs melhorando a performace e aumentando a escalabilidade (DUPONCHELLE, J.; GROSSMAN, J., 2017).

2.3.1. VMWARE FUSION

O VMware Fusion versão 10 é um software que permite virtualizar sistemas operacionais no macOS. Com ele podemos realizar os testes utilizando o emulador de rede GNS3, que possui uma versão que pode ser virtualizada. Essa versão é baseada no sistema operacional Ubuntu 14.04.5 LTS conhecida pelo codinome *Trusty*, a escolha pela arquitetura baseada em VM se deu devido a complexidade dos testes. (BOMBAL, DUPONCHELLE, 2017)

2.3.2. GNS3

O GNS 3, versão 2.1.6, é um software *open source* disponível no *github* e distribuído sobre a licença GNU v3.0 que permite a simulação de topologias redes em um ambiente virtualizado, compatível com Windows, MacOS e Linux. Com ele podemos criar e modificar topologias com grandes níveis de complexidade. Devido a sua similaridade com os equipamentos reais a sua utilização leva a uma redução de custos na montagem de um laboratório. Ele também suporta Dynamips, funcionalidade que permite emular um roteador e executar imagens com o IOS da Cisco. (DUPONCHELLE, J.; GROSSMAN, J., 2017).

O GNS3 possui uma flexibilidade muito grande, permitindo a emulação de vários *appliances* como Windows, Linux, Cumulus, Cisco, Arista, entre outros. Essa possibilidade permitiu que as aplicações necessárias para o desenvolvimento deste trabalho fossem testadas.

2.4. SISTEMA OPERACIONAL DE REDE

Durante a realização deste trabalho vários sistemas operacionais de rede foram analisados. Para a escolha alguns parâmetros foram definidos. Entre eles temos a utilização de um sistema que fosse baseado em Linux, que tivesse uma boa documentação para estudo, que fosse

possível a utilização no software de emulação como o GNS3 e que o sistema suportasse a aplicação em um ambiente de rede campus, com protocolos apropriados para este cenário. Assim foram selecionados alguns sistemas operacionais de rede, entre eles temos: Big switch, Pluribus, IP infusion, Pica8 e Cumulus.

O Big switch é uma companhia destinada a criação de soluções baseado em SDN para ambientes de *centro de dados*, ou seja, trabalhando com a desagregação do plano de controle e o plano dados com o gerenciamento feito através de um controlador. Seu sistema operacional Switch Light é baseado no Open Network Linux uma iniciativa da OPC que tenta padronizar um sistema que servirá de base para os demais sistemas operacionais de rede, assim tendo um repositório centralizado e reduzindo os esforços de engenharia (ONL, 2013). Sua documentação pode ser acessada no site do fabricante e seu sistema operacional pode ser emulado no GNS3. Porém conforme descrito anteriormente, devido a sua arquitetura ser baseada em SDN não foi possível a sua utilização.

O Pluribus é uma companhia focada na construção de soluções baseada em SDN e virtualização de rede, seu sistema operacional é o Open Netvisor Linux, o ONVL por padrão construí uma grande malha usando BGP e ECMP. Sua documentação está disponível ao público, porém o download do seu software só pode ser feito mediante a compra de suporte. Essa particularidade juntamente com a arquitetura do sistema operacional de rede ser baseada em SDN não se mostrou viável para a utilização deste projeto (PLURIBUS, 2015).

O IP infusion é empresa focada no desenvolvimento de software para grandes empresas e operadoras. Seu sistema operacional é o OcNOS, também baseado em Linux possui funcionalidades compatível com rede campus e funcionalidades de SDN sendo compatível com o protocolo OPENFLOW, permitindo assim uma compatibilidade com os dois tipos de cenários. O fabricante disponibiliza pouca documentação em seu site e é possível fazer o download do sobre por um período de 90 dias, entretanto não foi encontrado uma documentação que fosse demostrado com integrar o software OcNOS com o GNS3. (IP INFUSION, 2017)

A Pica8 é uma empresa focada no desenvolvimento de sistemas operacionais de rede para diferentes tipos de ambientes que vão de centro de dados até pequenas empresas, o PicOS é um software baseado em Linux que possui característica de redes campus e redes SDN. Sua documentação pode ser acessada no site de fabricante. Porém o download do sistema operacional apenas pode ser feito através da aquisição de suporte. (PICA8, 2017)

Por últimos temos o fabricante Cumulus, que possui um sistema operacional de rede compatível com a maioria de switches open networking do mercado. Sua arquitetura permite a

utilização em ambiente de rede campus, o cumulus não possui nenhuma compatibilidade com o SDN. Sua documentação é extensa e pode ser acessada. O fabricante também disponibiliza uma versão chamada Cumulus VX que pode ser virtualizada e integrada ao GNS3. Assim de acordo com as características e alinhamento com os objetivos deste trabalho o sistema operacional Cumulus VX se mostrou o ideal para os teste e integração com o emulador GNS3.

Quadro 1 - Comparativo entre os sistemas operacionais de rede

	Big switch	Pluribus	IP infusion	Pica8	Cumulus
NOS baseado em Linux	SIM Switch Light	SIM ONVL	SIM OcNOS	SIM PicOS	SIM Cumulus VX
NOS disponibilizado para testes	SIM	NÃO	TRIAL	NÃO	SIM
Documentação	SIM	Acesso Limitado	Acesso Limitado	SIM	SIM
Arquitetura	SDN	SDN	SDN e Ethernet	SDN e Ethernet	Ethernet

Fonte: Autor

2.4.1. CUMULUS

O Cumulus Linux versão 3.6.0, é um sistema operacional de rede baseado no Debian 8, também conhecido pelo codinome Jessie, especialmente construído para controlar as funções de um switch. Para o seu gerenciamento, possui um Linux Shell onde podem ser executados comandos com Nano ou Vi para fazer alterações nos arquivos de configuração, para os administradores que já estão acostumados com o gerenciamento de servidores não possui grandes diferenças, também possui uma interface de linha de comandos, o Cumulus Linux NCLU (Network Command Line Utility) com ele o administrador pode gerenciar o equipamento sem a necessidade de modificar diretamente os arquivos de configuração. Uma forma similar aos comandos CLI, permitindo uma fácil adaptação. Porém o NCLU não segue um padrão de mercado. (WELLS, 2018)

A arquitetura do Cumulus permite que ele gerencie os as funções software e hardware como se fosse um servidor. Aqui temos a principal diferença entre um servidor com várias

portas ethernet e um switch executando o Cumulus. Por possuir um hardware dedicado a comutação de pacotes de alta velocidade é baixa latência, conhecido com ASIC, o switch se torna superior ao servidor para as funções de rede. O ASIC é um componente que permite taxa de transferência que pode ser superior a 40 gigabit por segundo. Para comunicação do Cumulus com o ASIC dos equipamentos é utilizado um *daemon* chamado de swiched presente no Cumulus que permite abstrair a camada hardware do equipamento.

2.4.1.1. Switchd

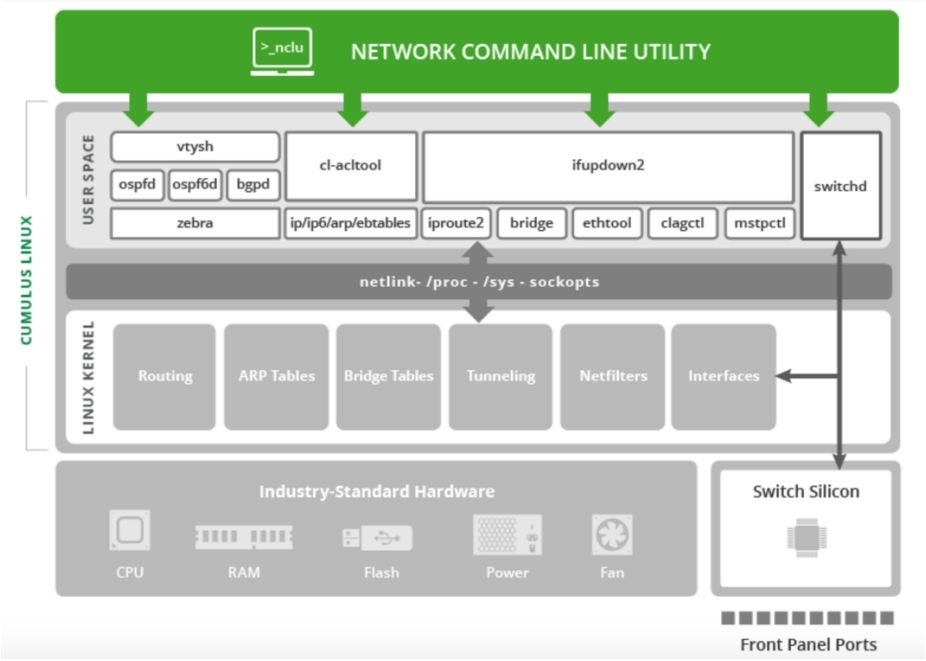
O switchd é um dos principais componentes do sistema operacional Cumulus, é o único componente do sistema que possui seu código fonte fechado. As propriedades intelectuais e patentes envolvidas na sua construção levou a necessidade de acordos de não divulgação entre a Cumulus e os fabricantes de *ASIC* como a Broadcom ou Mellanox (BRATACH, 2016). O daemon switchd é um dos principais responsáveis pela desagregação entre o hardware e software em um switch compatível com o open networking. Isso porque, ele possibilita que a comunicação entre o sistema operacional e *ASIC* ocorra. Quando alguma interação é necessária com o ASIC, a informações são enviadas pelo kernel utilizando o *socket*[1] netlink para o *daemon*[2] switchd. As informações recebidas são então convertidas para uma linguagem que pode ser entendida pelo ASIC. A comutação de pacotes e roteamento são alguns exemplos de funções controladas por este componente.

Como pode ser observado na Figura 6, temos um diagrama que mostra os elementos desta comunicação.

[1] Socket: Interface de programação deriado do Unix utilizados para a comunicação entre processos. (TANENBAUM, 2003)

[2] Daemon: São processo executados em plano de fundo. (TANENBAUM, 2003)

Figura 6 - Visão geral Network Command Line Utility



Fonte: Cumulus Networks, 2018.

2.4.1.2. FRRouting

O FRRouting é uma suíte com código aberto de protocolos de roteamento compatível com IPv4 e IPv6 para sistemas operacionais Linux e Unix. Ele foi derivado do projeto similar chamado Quagga. Ele é responsável pelas funções de roteamento relacionada aos protocolos BGP, IS-IS, LDP, OSPF, PIM e RIP. Para o controle dessas funções é utilizado um *daemon* chamado zebra, que controla os subprocessos que gerenciam as funções de roteamento, para o protocolo OSPF temos o *daemon* ospfd e para BGP temos o *daemon* bgpd. Essas funções são coordenadas interagindo com o kernel que por sua vez se comunica com o switchd mantendo a comunicação com o ASIC. Além do suporte nativo por parte do Cumulus, tem-se um outro ponto de destaque que é a interface chamada de vtysh que permite a configuração no padrão Cisco-Like o que traz ao administrador uma fácil adaptação. (FRROUTING, 2017)

2.4.1.3. /etc/network/interfaces

O arquivo /etc/network/interface é o local onde são realizadas as configurações relacionadas as interfaces do equipamento. Nele podemos definir vários parâmetros com se o endereçamento será fixo ou estático, criação de interface de loopback, o endereço IP, VLAN,

Agregação de links (BOND), Spanning Tree entre outros. A grande variedade de configurações que podem ser feitas neste arquivo e sua importância no correto funcionamento, leva a uma necessidade de automação neste processo (CUMULUS NETWORKS, 2018).

2.5. PROVISIONAMENTO

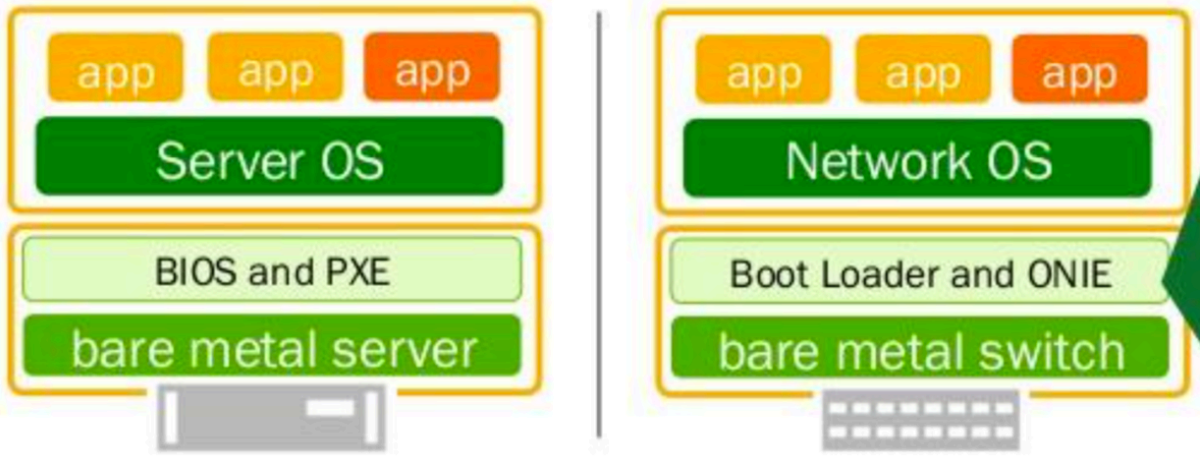
Provisionamento é um termo derivado do ambiente de virtualização, que significa que o equipamento tem a capacidade de se autoconfigurar após ser inserido em um ambiente com essa funcionalidade. Isto é muito importante, pois reduz o tempo de *downtime,* ou seja, o tempo que um equipamento fica inoperante na rede, e em casos onde não se tem uma mão de obra qualificada, o instalador apenas precisa conectar este equipamento na rede. Os equipamentos open networking possuem um pequeno sistema operacional pré-instalado denominado ONIE, ele tem a função de fornecer um ambiente para o provisionamento automático para o sistema operacional de rede ou possibilidade de alteração, pode-se fazer uma correlação com o funcionando do PXE boot ou U-boot nos servidores, conforme destacado na Figura 7 (ONIE, 2018).

O ONIE atua antes que o sistema operacional de rede esteja instalado em um equipamento, tornando o processo de instalação ou atualização de switch muito mais simples. Outra forma de provisionamento que o Cumulus oferece é o processo denomina Zero Touch provisioning ou ZTP permite provisionamento de configurações no momento de inicialização do equipamento.

2.5.1. ONIE

Baseado na nova arquitetura proposta pelo *open compute project*, em que o engenheiro de rede passa a ter capacidade de definir o sistema operacional que deseja para seu equipamento, bem como suas funcionalidades, uma ferramenta tem papel fundamental neste empoderamento: o ONIE (Open Networking Install Environment). O ONIE é um software baseado em LINUX que atua como *boot loader* entre o *hardware* do switch de rede e o sistema operacional desejado. O ONIE permite que um switch open networking carregue o sistema operacional de rede escolhido. Na Figura 7, ele está presente como uma ferramenta intermediária entre hardware e o software desejado.

Figura 7 - Comparativo servidor bare metal e switch bare metal

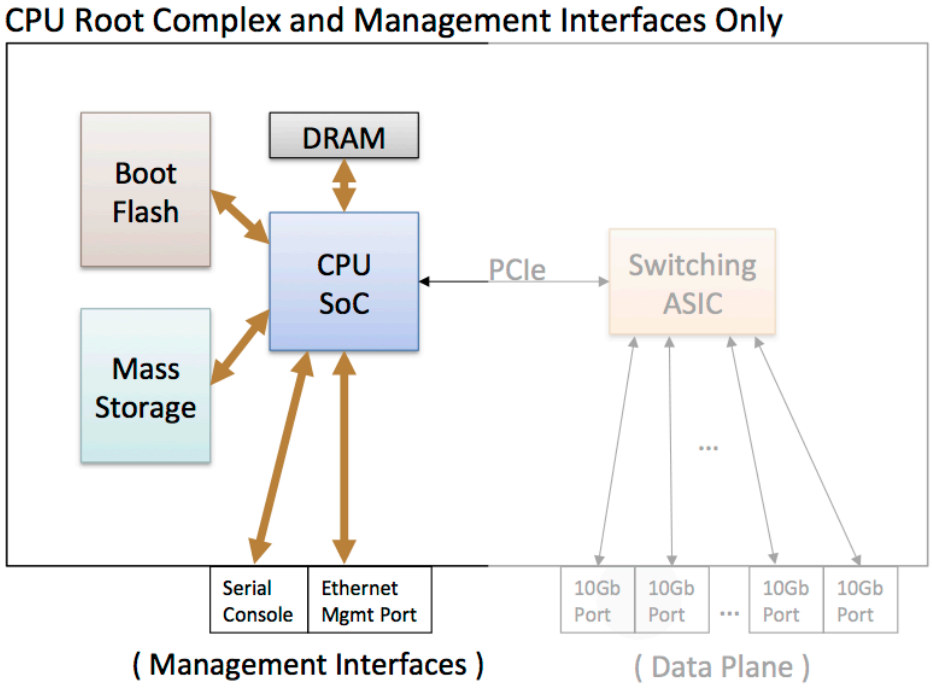


Fonte: LAMA, 2016.

O ONIE utiliza somente a CPU Root Complex do switch, este elemento é composto pelas CPU, memórias e Interfaces de gerenciamento não interagindo com as interfaces controladas pelo o ASIC, conforme ilustrado na Figura 8.

Outra característica que levou a esta arquitetura é que caso fosse necessário a utilização do ASIC para essa funcionalidade seria necessária o pagamento de royalties, então os projetistas do ONIE optaram por uma arquitetura aberta e gratuita (ONIE, 2018).

Figura 8 - Visão geral ONIE

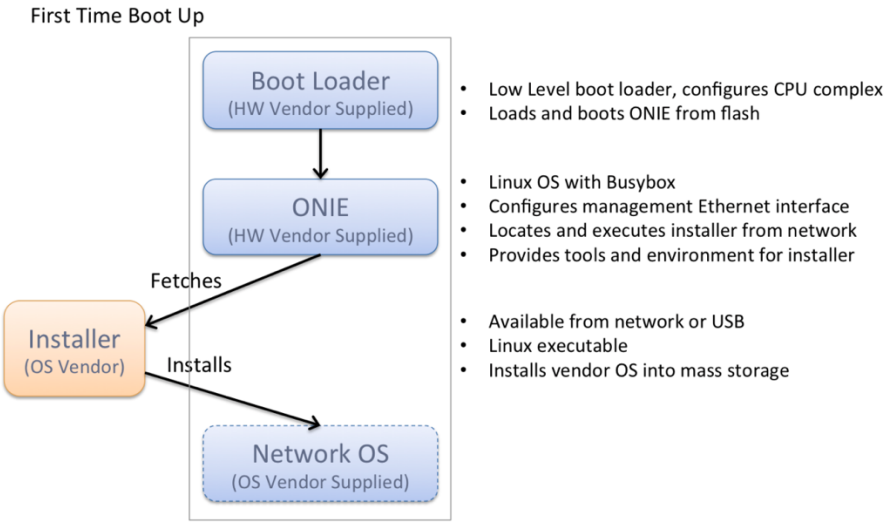


Fonte: ONIE, 2018.

Antes que o ONIE seja executado existe um componente essencial é o processo de boot do equipamento. Ele é iniciado assim que o equipamento é alimentado. A primeira etapa é executada pelo U-boot (Universal Boot Loader). Ele é um *open source* firmware compatível com várias plaformas que tem a função de configurar os contadores de memória e SDRAM (Synchronous Dynamic RAM) possibilitando a alocação e execução do ONIE em uma área de memória. (U-BOOT, 2018)

Após a alocação do ONIE pelo U-boot, inicia-se uma verificação buscado identificar se o equipamento já possui algum sistema operacional de rede, caso não seja identificado, é iniciado uma busca pelas informações sobre o hardware de equipamentos para que serão repassadas ao instalador. Através do instalador é possível a escolha do sistema operacional de rede que será instalado no equipamento. O ONIE suporta de descoberta através de IPv6 e IPv4 utilizando o campo options do DHCP, HTTP, FTP, TFTP e USB (ONIE, 2018). Todo este processo é ilustrado na Figura 9.

Figura 9 - Esquema de execução do ONIE



Fonte: ONIE, 2018.

É importante ressaltar que nem todos os switches de rede suportam o ONIE, impossibilitando a alteração dos sistemas operacionais rede. O suporte ao ONIE caracteriza um equipamento como open networking.

2.5.2. Zero Touch Provisioning – ZTP

O *Zero Touch Provisioning* permite que os equipamentos de rede possam ser rapidamente configurados em um ambiente, reduzindo o tempo de implantação ou substituição gerando uma economia de tempo e recursos. Na primeira inicialização, o Cumulus executará o ZTP para que o equipamento possa ser provisionado.

A rotina de execução do provisionamento é iniciada através da interface de gerenciamento verificando se o equipamento possui algum *script* local, caso não seja encontrado, o equipamento passa a verificar se a interface USB possui algum *script*, por fim, o ZTP verifica se o servidor DHCP possui algum endereço dentro do campo options 239, essa opção é verificada para encontrar um caminho para download do *script* de provisionamento. Uma informação que deve conter no *script* é a flag CUMULUS-AUTOPROVISIONING e o script deve retornar o valor 0 ao final da sua execução. Os *scripts* podem ser escritos utilizando uma variedade de linguagens como Perl, Pyton, Ruby e Shell. Pode-se também em alguns casos como o Puppet ou o Chef automatizar a instalação de agentes e outras configurações. No caso do Ansible, uma vez que está ferramenta não requer a instalação de agente será demostrado o provisionamento criando o usuário “cumulus” e instalação das chaves para o SSH.

2.6. AUTOMAÇÃO

A automação em um ambiente de TI consiste em utilizar software para criar instruções ou processos reproduzíveis capazes de substituir ou reduzir a interação humana com os sistemas. Ou seja, a partir de instruções previamente estabelecidas o software de automação é capaz de reduzir ou eliminar a interversão humana. O intuito da automação é a redução ou eliminação de tarefas repetitivas ou manual que consome o tempo das equipes de TI. (REDHAT, 2018).

A virtualização trouxe uma flexibilidade muito grande para os ambientes de TI, termos como provisionamento e automação substituíram os processos manuais e repetitivos, reduzindo a quantidade de erros e aumentando a qualidade de entrega. Tudo isso atrelado a uma redução das atividades executadas pelas equipes de TI. A grande responsável por essa mudança foi o surgimento de metodologias ágeis que buscavam integrar diferentes áreas como de desenvolvimento e operação. Assim surgiu o DEVOPS. (REDHAT, 2018).

2.6.1. DEVOPS

O DEVOPS é um movimento cultural que busca melhorar a comunicação entre a área desenvolvimento e operações ou infraestrutura de TI. O objetivo é que desenvolvedores e engenheiros busquem uma integração eliminando tarefas rotineiras através da automação, com resultado tem-se uma melhoria na qualidade e agilidade na entrega de projetos. Essa aproximação altera a antiga dinâmica que visualizava essas áreas como diferentes grupos. Muitas vezes por essa falta de comunicação gerava um insucesso nos projetos. (SHARMA, COYNE, 2017)

2.6.2. NETDEVOPS

A utilização da cultura DEVOPS no ambiente de rede cria um novo termo, conhecido com NETDEVOPS (Network, Development and Operations). Assim como o DEVOPS o NETDEVOPS busca uma harmonização de processos e pessoas. (BENOKRAITIS, 2018)

O Cumulus é um sistema operacional de rede baseado em Linux, isso traz a possibilidade de o switch passar a ser gerenciado com um servidor, utilizando ferramentas de automação com o Ansible. As configurações em um ambiente de redes muitas vezes são complexas e repetitivas, um endereçamento errado pode deixar toda uma rede inoperante. Com a automação temos a possibilidade de fazer o gerenciamento das configurações, podendo utilizar o versionamento, registrando todo o histórico das alterações, trazendo uma melhoria na agilidade e escalabilidade das operações diárias.

2.6.3. FERRAMENTAS DE AUTOMAÇÃO

Para a escolha da ferramenta de automação foram selecionadas algumas ferramentas, entre elas temos: Puppet, Chef, Saltstack e Ansible. A escolha foi baseada nos parâmetros relacionada a tipo de linguagem, arquitetura, simplicidade, existência de módulo de integração e compatibilidade com o Cumulus.

O Puppet é uma das ferramentas de automação mais conhecidas, ela é escrita em Ruby é possui uma arquitetura baseado em cliente e servidor, para inserção de comando é utilização um agente que deve ser instalado no cliente. Essa ferramenta possui módulos compatíveis com o Cumulus, pórem sua utilização requer grandes conhecimentos programação ser comparado com o Ansible.

O Chef é uma ferramenta escrita em Ruby que possui uma arquitetura baseada em cliente e servidor, seu mecanismo de inserção de comandos requer um agente instalado no cliente, ou seja, baseado em *pull*[3]. Os arquivos de configuração são chamados de cookbooks e são escritos Ruby o que requer conhecimento nesta linguagem. O Chef possui módulos de integração com o Cumulus. Entretanto sua utilização mais ampla é feita por equipes que já dominam a linguagem Ruby.

O Saltstack é uma ferramenta escrita em python, que possui uma arquitetura que funciona de duas formas, um baseado em *push*[3] e outra em *pull* de comandos. Esta ferramenta é se assemelha do software Ansible, porém não apresenta uma compatibilidade com o NCLU do Cumulus. Sua utilização é mais indicada para equipamentos como Cisco e Juniper.

O Ansible foi escolhido devido sua simplicidade na sua utilização, uma vez que não necessita de um agente instalado no dispositivo, sua característica no mecanismo de inserção de comandos baseado em *push* mostra-se bastante efetivo em um ambiente de rede campus onde a quantidade de equipamentos é limitada. Sua linguagem baseada em arquivos YAML (YAML - Ain't Markup Language) é utilizada para criação dos *playbooks*[3] e possui módulo compatível com o NCLU (Network Command Line Utility), não requer grandes conhecimento de programação sendo necessário apenas passar os parâmetros adequados para o playbook. Essa ferramenta possui módulos compatíveis com o Cumulus Linux.

Quadro 2 - Comparativo entre as ferramentas de automação.

	Puppet	Chef	Saltstack	Ansible
Linguagem	Ruby	Ruby	Python	Python
Arquitetura	Master/Agente	Master/Agente	Master/Agente	Master/Agente
Mecanismo de configuração[4]	Pull	Pull	Pull e Push	Push
Simplicidade	Requer conhecimento em programação para construção do Manifesto	Requer conhecimento em programação para construção do	Utiliza YAML para construção dos módulos	Utiliza YAML para construção do Playbook

[3] Playbook, que consiste no script das tarefas a serem executadas geralmente escritas em YAML. (MENZEN,2015)

[4] Mecanismo de configuração: Existem dois métodos, no primeiro o servidor realiza a requisição é conhecido como Push e comumente é utilizando um agente e o segundo e quando o cliente realiza a requisição não necessitando de instalação de uma agente. (CHO, K. C. S. J. 2007)

	(Ruby DSL)	Cookbook (Ruby DSL)		
Existência de módulo de integração	SIM	SIM	SIM	SIM
Compatibilidade com o Cumulus	SIM	SIM	SIM	SIM

Fonte: RAZA, 2016 com adaptações.

2.6.3.1. ANSIBLE

O Ansible é uma ferramenta de código aberto que automatiza os processos de configuração. Ela foi inicialmente desenvolvida em python por Michael DeHann em 2012 (MENZEN, 2015). Sua arquitetura possibilita uma fácil implementação, diferentemente de outros softwares de gerência de configuração como o Puppet, Chef e Saltstack. O Ansible não requer nenhuma instalação de agente no cliente. Para que o servidor de configuração acesse o cliente é utilizado uma conexão segura através do protocolo SSH. Desta forma, o software faz um *push* de comandos no cliente, essa técnica permite uma execução imediata dos comandos. Porém requer uma intervenção do administrador.

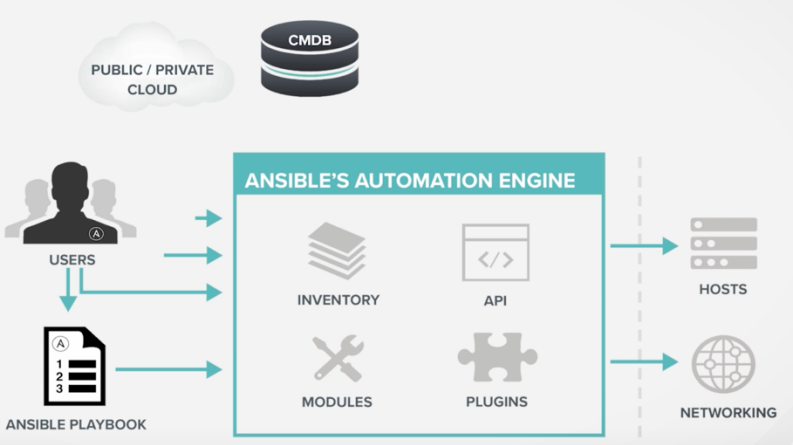
A tarefa de administrar equipamentos de rede requer conhecimento e planejamento. Para sua instalação geralmente são gastos muitas horas de implementação e tarefas manuais. Muitas vezes são utilizados cabos seriais, que devem ser conectados fisicamente ao equipamento, o que impossibilita uma implementação automatizada e remota.

O Ansible tenta resolver alguns destes problemas, em conjunto com outras ferramentas, e é possível provisionar um equipamento, enviar configurações, e após isso passar a gerenciar suas configurações de forma automática.

A Figura 10 a seguir demostra o funcionamento do Ansible. Essa é uma plataforma de automação que possui uma linguagem baseada em *playbooks* para entradas de comandos, essas entradas devem ser repassadas pelos usuários. Os *playbooks* são escritos em uma linguagem chamada de YAML, essa linguagem pode ser lida é não requer conhecimentos de programação para sua utilização, apenas devem ser passados os parâmetros corretos, simplificando a sua utilização. O *playbook* é executado sequencialmente de cima para baixo, e através dele podem ser invocados os módulos. Os módulos são partes de código que tem a função de automatizar algumas tarefas. São desenvolvidos por uma comunidade e são escritos em qualquer

linguagem. Outro elemento importante na utilização do Ansible é o *Inventory*, nele são descritos todos os elementos ou grupos que serão gerenciados. Através desta combinação de elementos, ações podem ser automatizadas e inseridas em diversos equipamentos utilizando SSH. (PERZ, 2017)

Figura 10 - Visão geral Ansible



Fonte: PERZ, 2017.

CAPÍTULO 3 – PROPOSTA DE SOLUÇÃO DE AUTOMACÃO E PROVISIONAMENTO DE UMA REDE CAMPUS.

Neste capítulo do trabalho será apresentado a proposta de solução para automação e provisionamento de uma rede campus. Também será detalhado como as ferramentas foram escolhidas e como serão utilizadas para a configuração do ambiente de testes relacionados ao provisionamento, gerenciamento e automação de rede.

3. LEVANTAMENTO BIBLIOGRÁFICO

Para o referenciamento desse trabalho, foram obtidos dados e informações de diferentes meios como livros, artigos e outros tipos de trabalhos acadêmicos, sites especializados e site de fabricantes de equipamentos de rede e software de automação também foram consultados. Assuntos como SDN, open networking, OCP, DEVOPS, NETDEVOPS, designer de rede, softwares de automação e provisionamento foram os principais temas desta pesquisa.

3.1. ANÁLISE DO AMBIENTE

O ambiente escolhido para os testes neste trabalho foi uma rede campus de médio porte. Essas redes comumente possuem diversos serviços, uma exigência de tráfego e necessidade de disponibilidade, isso alinhado a um ambiente onde alterações de configurações são constantes, mostrou-se ideal para aplicação dos conceitos e hipótese deste trabalho. Assim uma rede onde os novos equipamentos podem ser provisionados automaticamente e as tarefas rotineiras pode ser automatizada traz um benefício tanto para as intuições quanto para os funcionários. Uma vez que a utilização de equipamentos open networking pode trazer uma redução de até 53% no custo de aquisição dos equipamentos. (PICA8, 2017)

Isso agrupado a uma redução do tempo de manutenção e instalação destes ativos, possibilita que as equipes tenham mais tempo para realização de outras tarefas. Essa evolução no gerenciamento dos ativos de rede permite que as equipes de rede estejam alinhadas com as mais recentes técnicas de gerenciamento de rede e possibilitando assim uma fácil migração para as redes baseadas em software ou SDN.

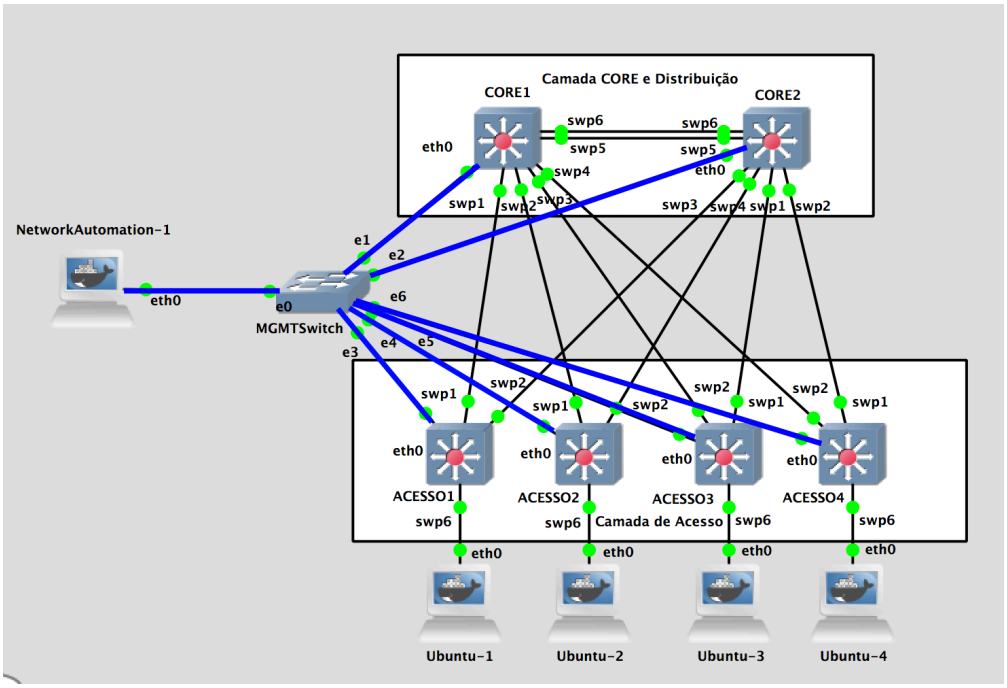
3.2. TOPOLOGIA PROPOSTA

Neste trabalho, é considerado um ambiente que possui uma hierarquia de duas camadas composta por dois switches na camada de núcleo/distribuição, utilizando um protocolo de alta disponibilidade. Para a camada de acesso são conectados quatro switches conectados à camada de núcleo/distribuição utilizando um protocolo de redundância de links, esse protocolo é alternativo ao STP, não desabilitando nenhum caminho e aumentando a largura de banda. Considerando que os equipamentos da camada Core são equipamentos de 48 portas essa rede suportaria a expansão até 48 equipamentos na camada de acesso provendo conectividade para até 2300 pontos de rede considerando equipamentos de 48 portas na camada de acesso.

Também foram projetados links redundantes para conexões com outros ambientes com de *Datacenter, WAN* ou Internet. Paralelo a rede de usuários foi criada uma rede para gerenciamento e provisionamento dos switches utilizando o ONIE essa rede está destacada em azul. Essa necessidade provém das limitações do ONIE poder ser executado apenas na interface de gerenciamento.

Um servidor foi conectado a rede de gerência para as funções de provisionamento e automação.

Figura 11 - Topologia de rede proposta para ambiente de testes



Fonte: Autor.

3.3. AMBIENTE DE TESTES

Para a estruturação do ambiente de testes foi utilizado a ferramenta de emulação GNS3 integrado com o VMware Fusion. Nele foi instalado uma versão do sistema operacional de rede Cumulus em sua versão virtualizada conhecida com Cumulus VX, todas as configurações de camada 2 foram feitas através de arquivo /etc/network/interfaces ou utilizando a NCLU do Cumulus, já para as configurações de camada 3 foram feitas o FRRouting. Também foi instalado um servidor de automação, onde foi instalado o Ansible, um DHCP Server e Servidor Web. A topologia foi montada e as interfaces de rede foram conectadas de acordo com a Figura 11. A rede de gerenciamento destacada é azul conecta todos os switches ao servidor de automação.

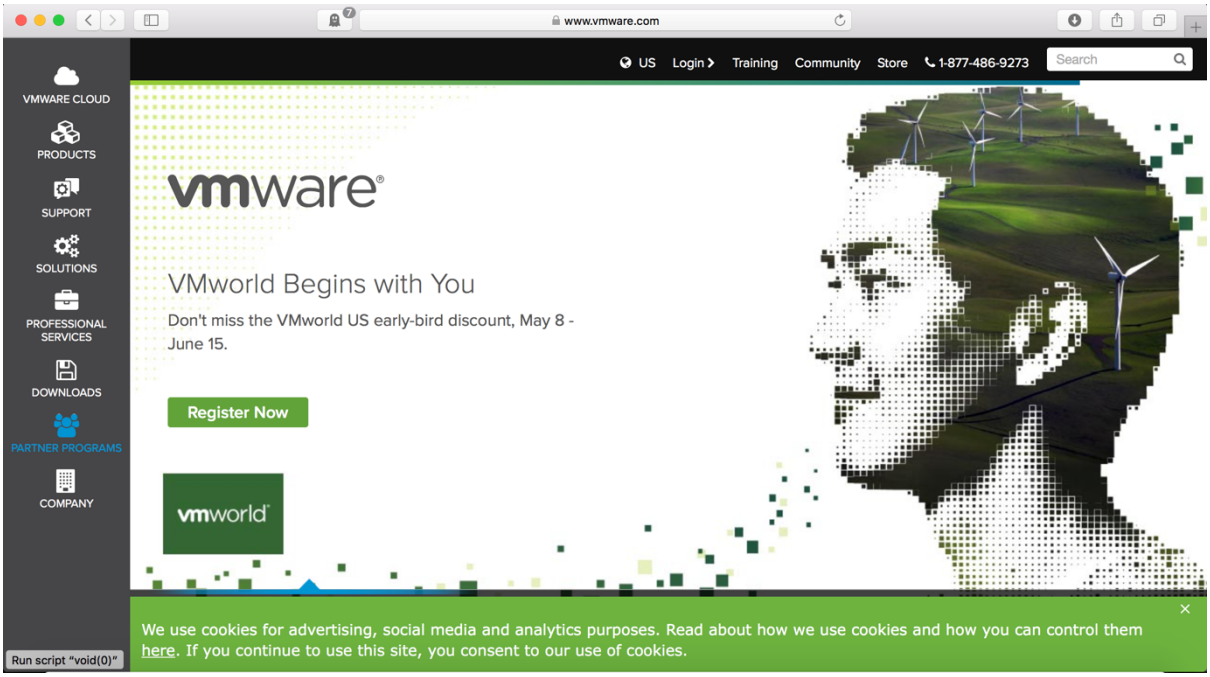
No item 3.3.1 serão mostrados como as ferramentas foram instaladas e integradas para a construção do ambiente de teste.

3.3.1. INSTALAÇÃO SOFTWARE DE EMULAÇÃO

Para instalação e configuração do software do emulação GNS3 e integração com o ferramenta de virtualização VMware Fusion no sistema operacional MacOS, utilizou-se uma documentação disponível no site https://docs.gns3.com/1wdfvS-OlFfOf7HWZoSXMbG58C4pMSy7vKJFiKKVResc/index.html#h.seowsyt9e4j7 . A seguir os passos para a instalação destes softwares.

1˚ Passo – Acessar o site www.vmware.com, conforme a Figura 12.

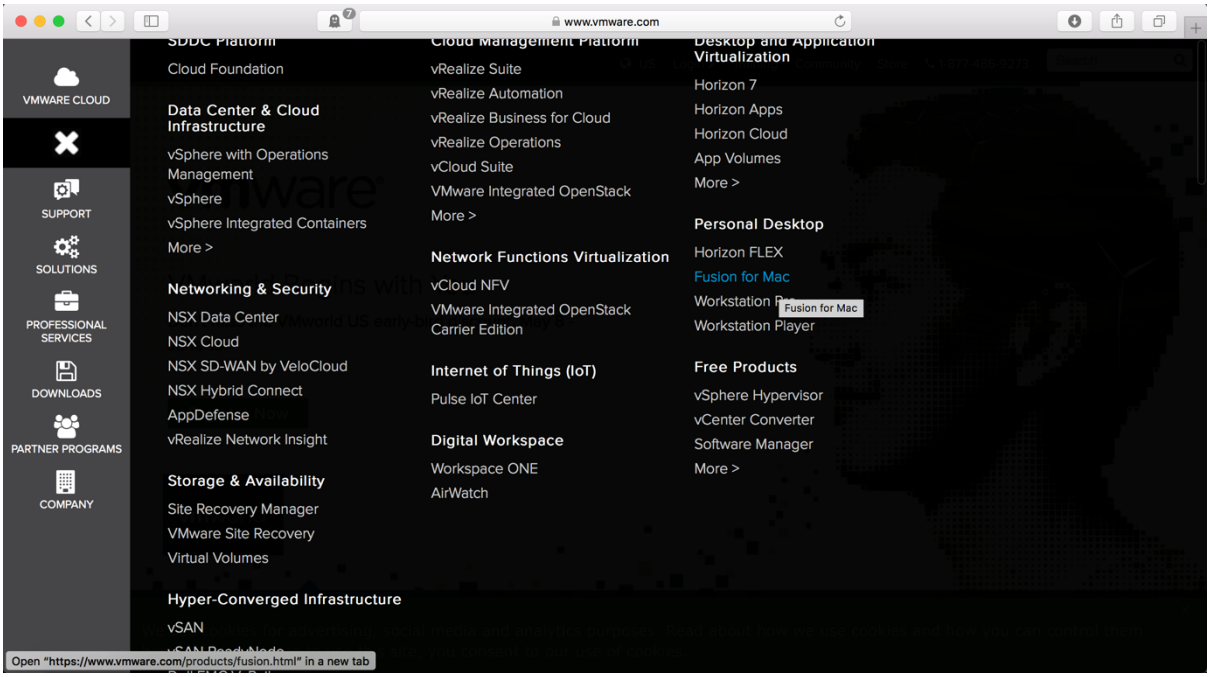
Figura 12 - Acesso ao site da VMware



Fonte: Autor.

2˚ Passo – Clicar no item downloads e selecionar a opção Fusion For Mac; , conforme a Figura 13

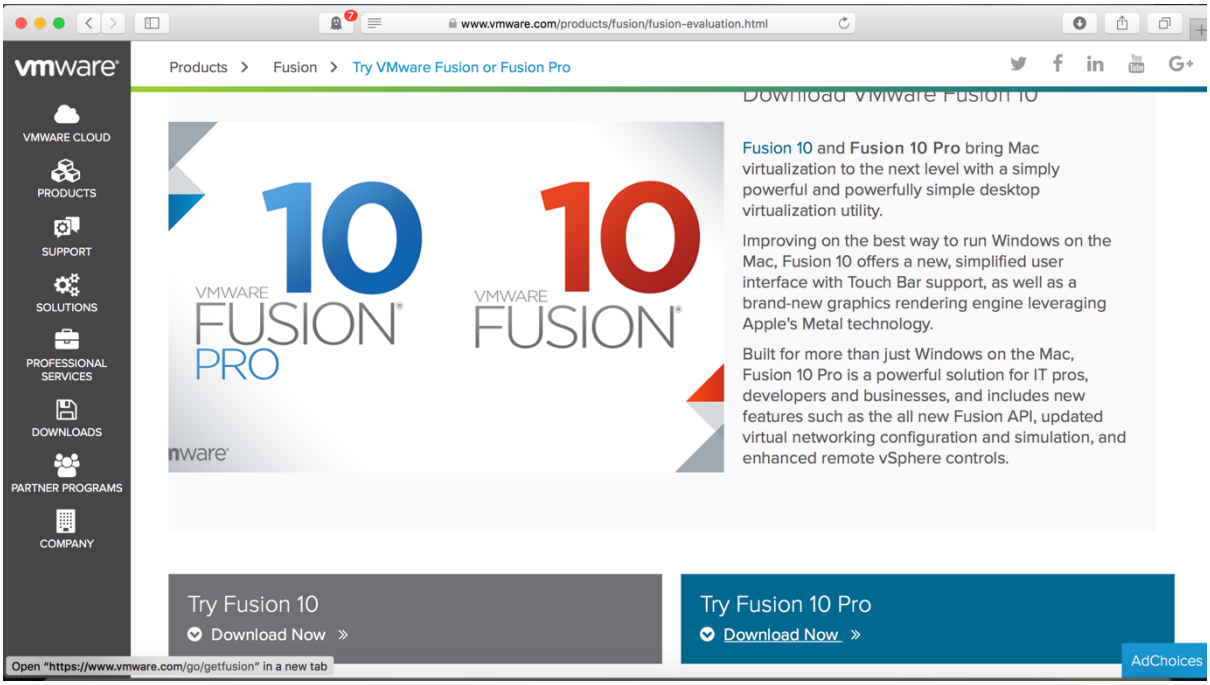
Figura 13 - Download VMware Fusion.



Fonte: Autor

3° Passo – Clicar no item Try Fusion Pro, este download baixa uma versão de demonstração válida por 30 dias, conforme a Figura 14.

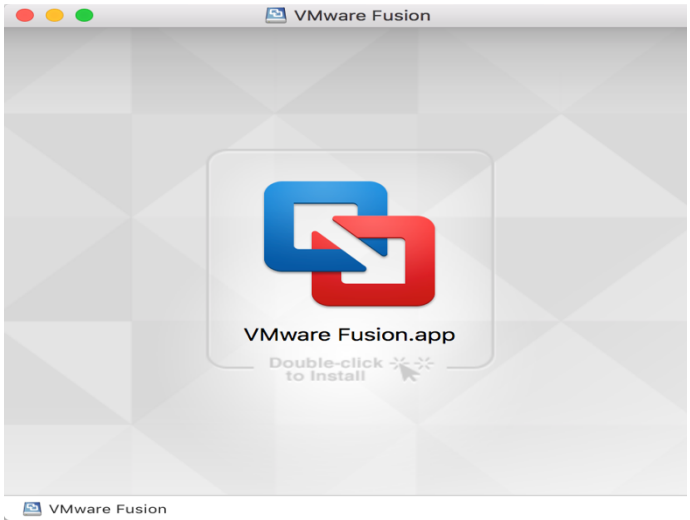
Figura 14 - Download VMware Fusion



Fonte: Autor

4° Passo – Após o download do arquivo, deve-se instalar o programa, conforme a Figura 15.

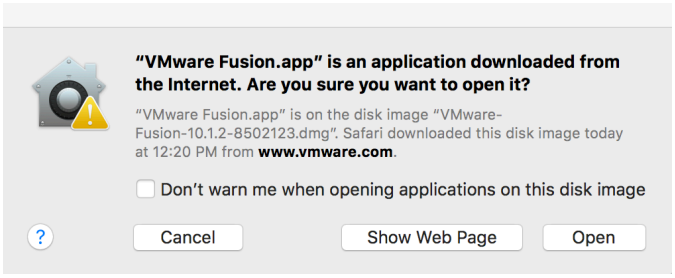
Figura 15 - Instalação VMware Fusion no Mac



Fonte: Autor

5° Passo – Após iniciar a instalação. Aparecerá um aviso dizendo que este arquivo foi baixado pela Internet e perguntará se você quer realmente instalar; Clique em Abrir ou Open, conforme a Figura 16.

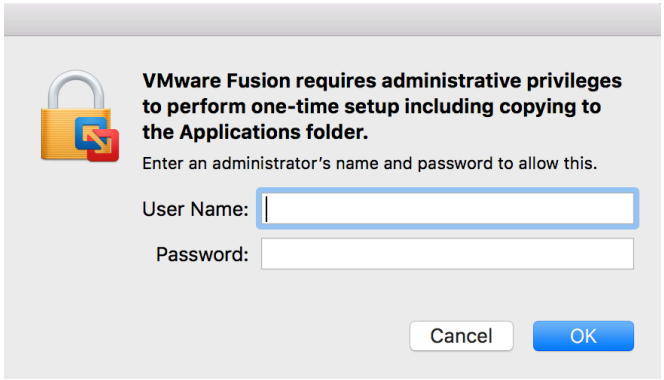
Figura 16 - Aviso durante instalação do VMware Fusion no Mac



Fonte: Autor

6° Passo – Entre com as credenciais do sistema, conforme a Figura 17.

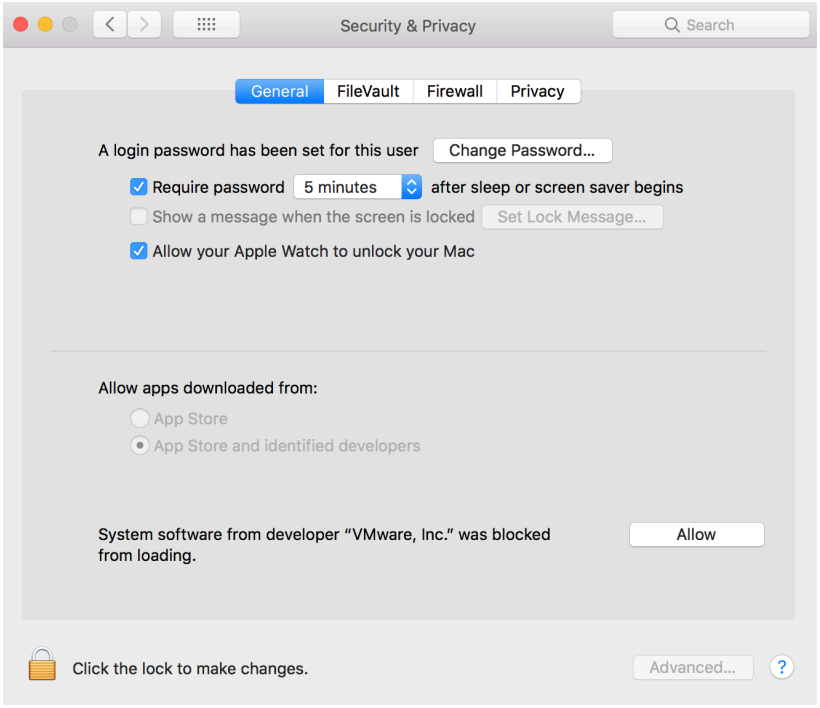
Figura 17 - Tela para inserção das credencias no Mac



Fonte: Autor

7° Passo – Após esta etapa, deve ir nas preferências do sistema, segurança & privacidade; e permitir a execução do programa, conforme a Figura 18.

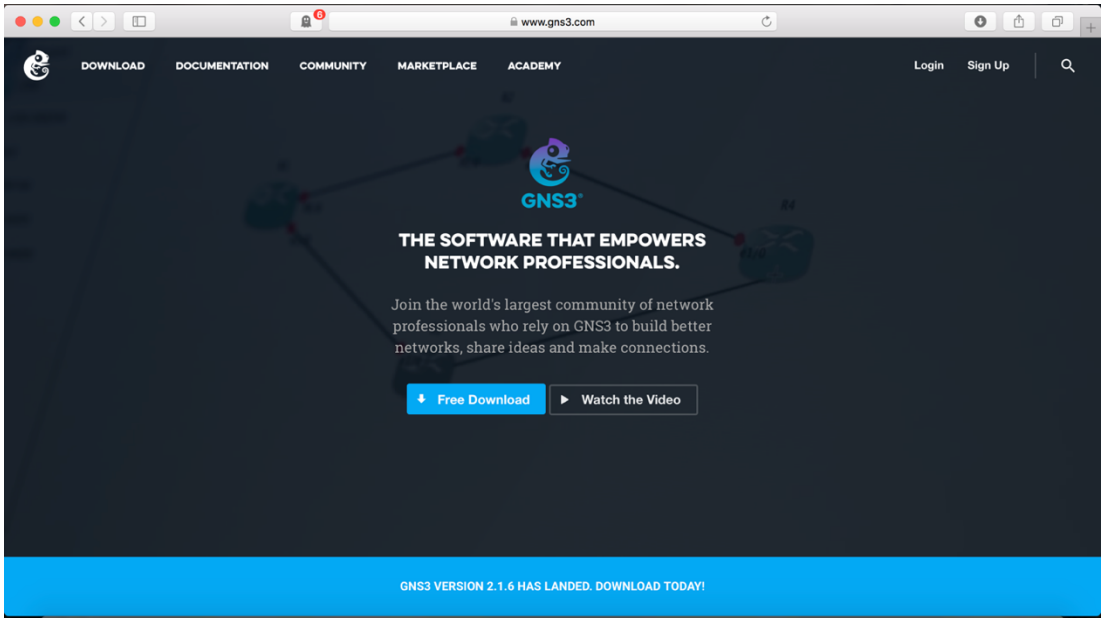
Figura 18 - Configuração para execução do VMware Fusion no Mac.



Fonte: Autor

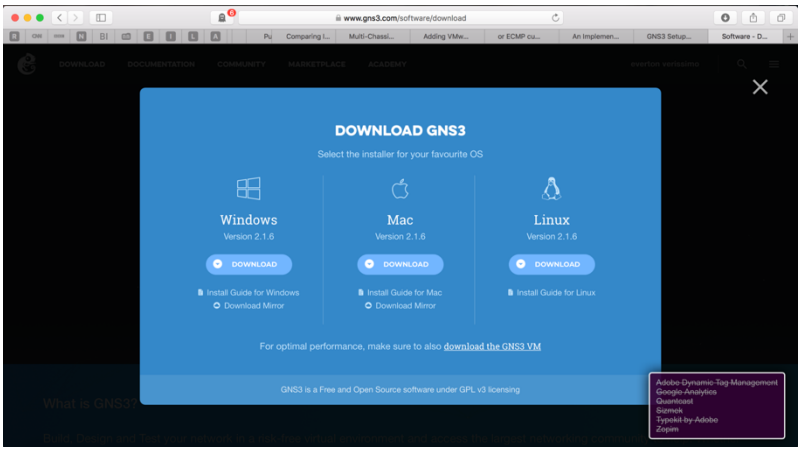
8° Passo – Após esta etapa, deve-se acessar o site www.gns3.com, clicar na aba Downloads, conforme a Figura 19.

Figura 19 - Acesso ao site do GNS3



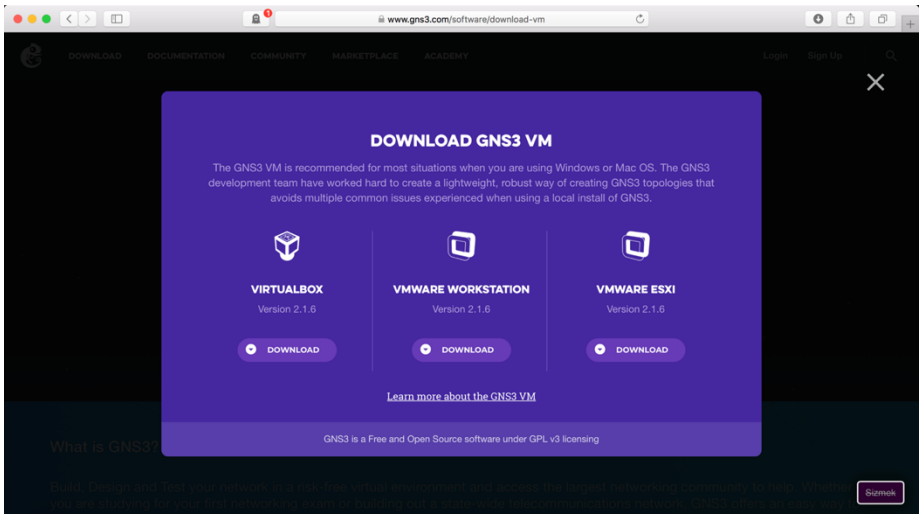
9˚ Passo – Após esta etapa, fazer os downloads dos arquivos GNS3 2.1.6 para Mac e GNS3 VM (Virtual Machine) o VMware workstation que também e compatível com o VMware Fusion, conforme as Figuras 20 e 21.

Figura 20 - Tela para download do software GNS3.app



Fonte: Autor

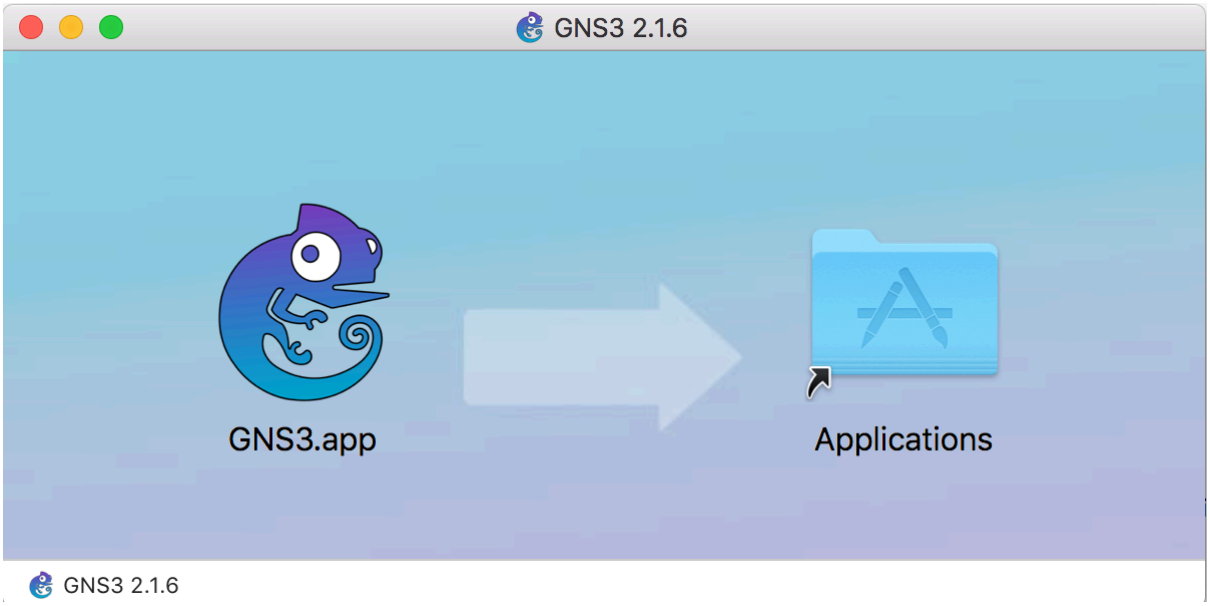
Figura 21 - Tela para download GNS3 VM



Fonte: Autor

10˚ Passo – Após esta etapa, deve ser feita a instalação do GNS3 2.1.6 para Mac; Sendo necessário apenas clicar em GNS3.app e arrastar para pasta Applications, será solicitado as credenciais do sistema e após este passo estará instalado o GNS3.app, conforme a Figura 22.

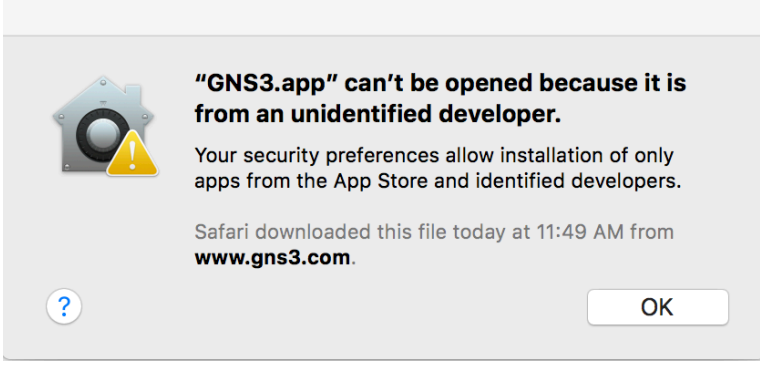
Figura 22 - Instalação do software GNS3



Fonte: Autor

11˚ Passo – Na primeira execução do GNS3.app aparecerá um aviso dizendo que o arquivo não pode ser aberto devido o desenvolvedor são ser identificado; Clique em OK, conforme a Figura 23.

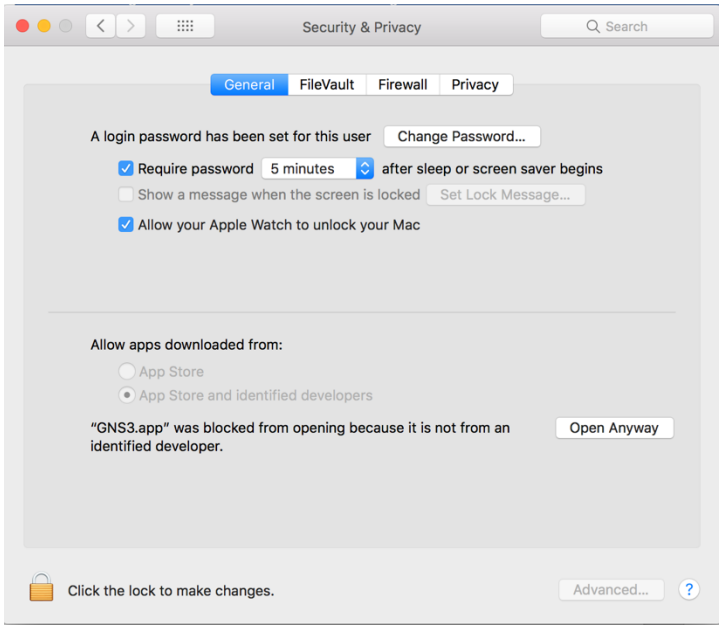
Figura 23 - Aviso instalação Mac para o GNS3.app



Fonte: Autor

12˚ Passo – Entre com as credenciais do sistema, após essa etapa deve ir nas preferências dos sistemas, segurança & privacidade; e permitir a execução do programa, conforme a Figura 24

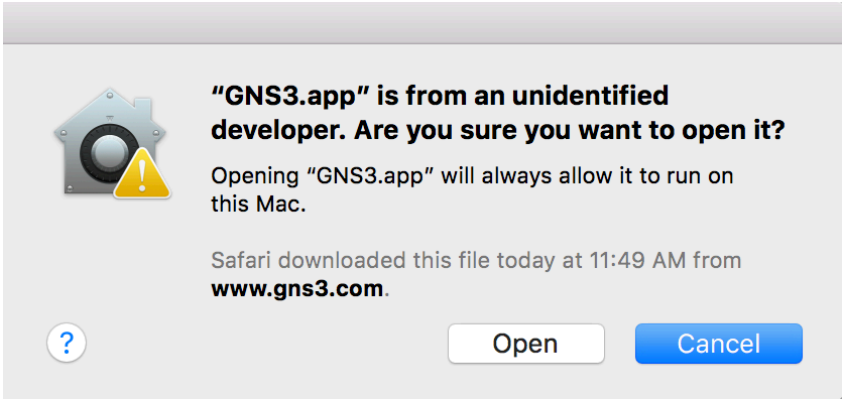
Figura 24 - Configuração para execução do GNS3.app no Mac.



Fonte: Autor

13˚ Passo – Entrar com as credencias do sistema, conforme a Figura 25.

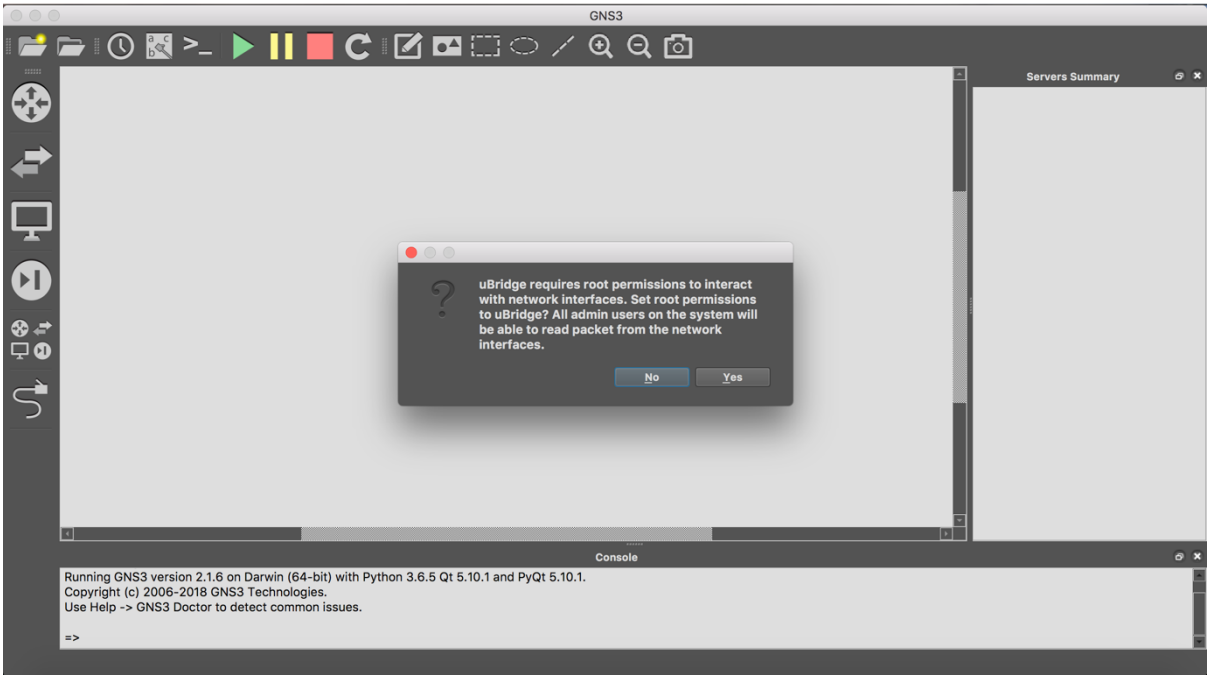
Figura 25 - Permissão para execução do software GNS3.app



Fonte: Autor

14˚ Passo – Entrar com as credencias do sistema, e o programa será executado; na primeira inicialização do programa aparecerá uma mensagem para configuração do uBridge (essa ferramenta permiti a criação e integração entre tecnologia usando UDP, Ethernet, interfaces do sistema) sua execução requer o acesso root, conforme a Figura 26.

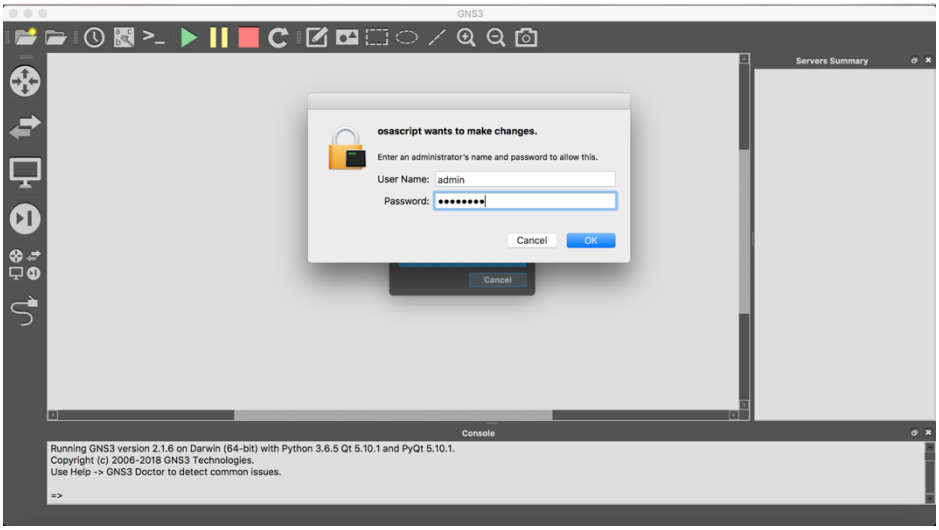
Figura 26 - Primeira inicialização do software GNS3 e Instalação uBridge



Fonte: Autor

15˚ Passo – Após o passo anterior, aparecerá mais uma janela solicitando a execução de um script para configuração do sistema, conforme a Figura 27.

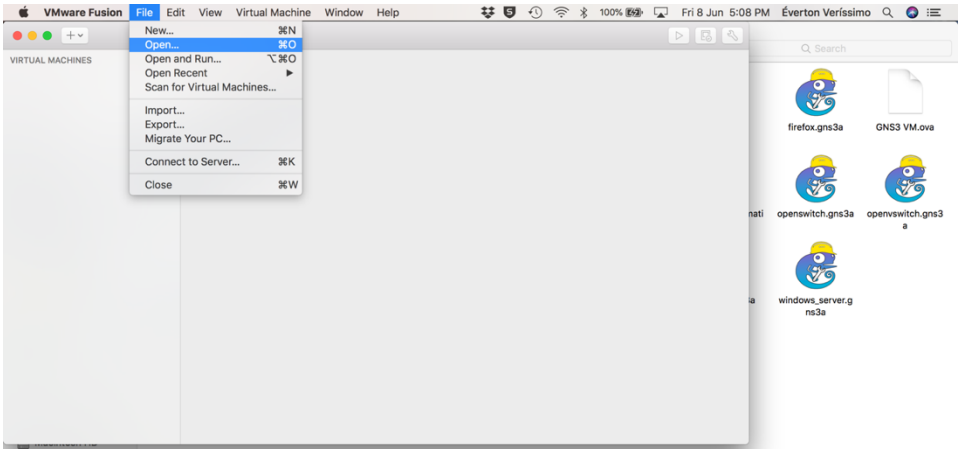
Figura 27 - Permissão para execução do script



Fonte: Autor

16˚ Passo – Agora precisa-se apenas integrar o GNS3.app ao GNS3 VM, para isso é utilizado o VMware Fusion; Após abrir o VMware Fusion é necessário importar o arquivo GNS3VM, conforme a Figura 28.

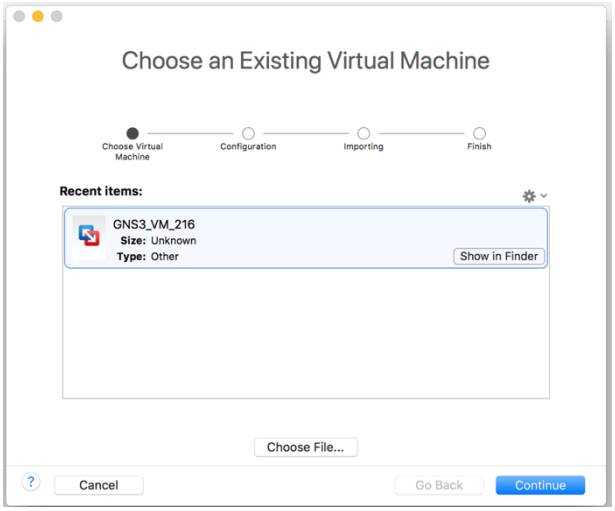
Figura 28 - Instalação no GNS3 VM no VMware Fusion



Fonte: Autor

17˚ Passo – Após este passo, será iniciado um assistente de importação, conforme a Figura 29.

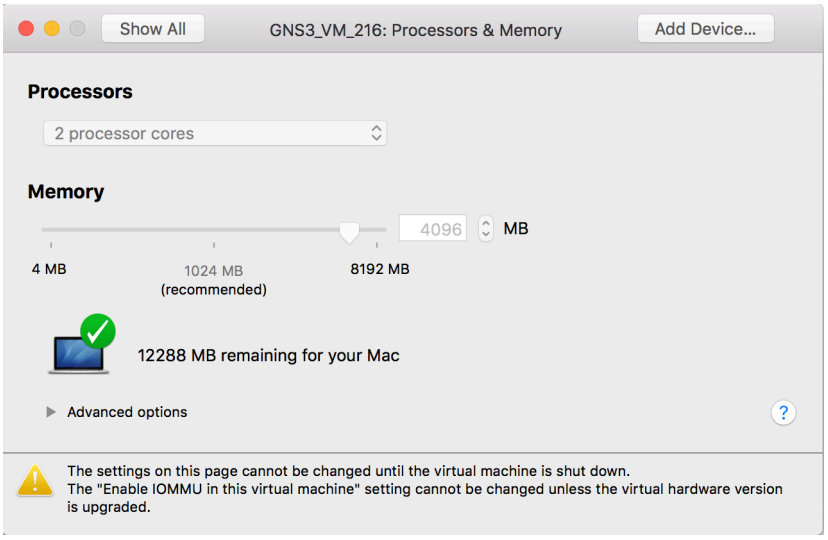
Figura 29 - Importação no GNS3 VM no VMware Fusion



Fonte: Autor

18° Passo – É recomendado o ajuste nas configurações da VM, utilizando 2 ou mais cores do processador e 4096 MB ou mais de memória, conforme a Figura 30.

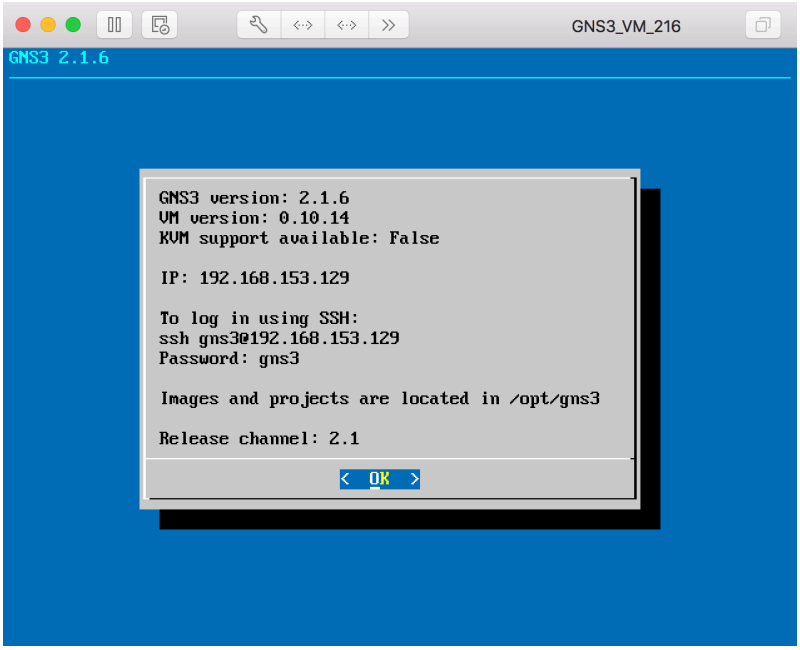
Figura 30 - Configuração de processador e memória da VM



Fonte: Autor

19° Passo – Após essa etapa a o GNS3 VM será iniciado, porém inicialmente ele não terá o suporte ao modo KVM que permite um melhor desempenho do sistema, conforme a Figura 31.

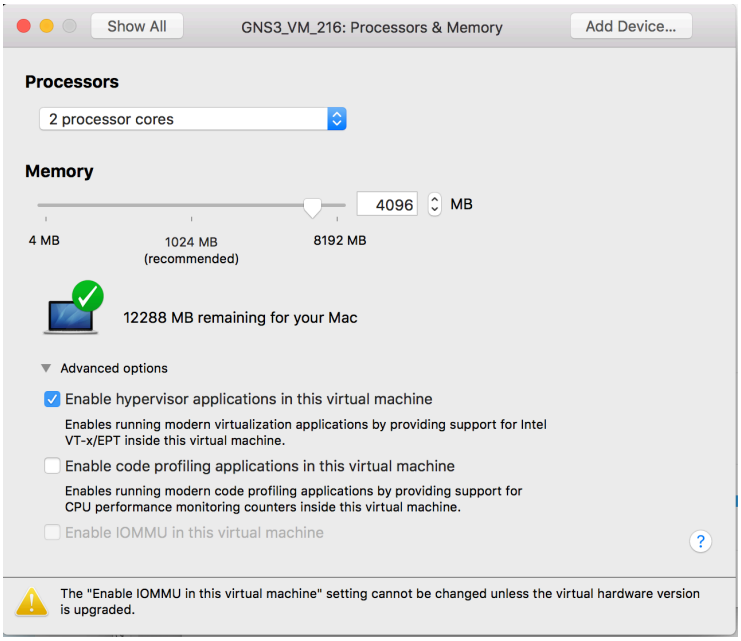
Figura 31 - GNS3 VM funcionando sem suporte a KVM



Fonte: Autor

20˚ Passo – Para que este suporte seja habilitado, deve-se desligar a VM, ir nas configurações de processador e memória da VM e habilitar o suporte ao VT-x/EPT, conforme a Figura 32.

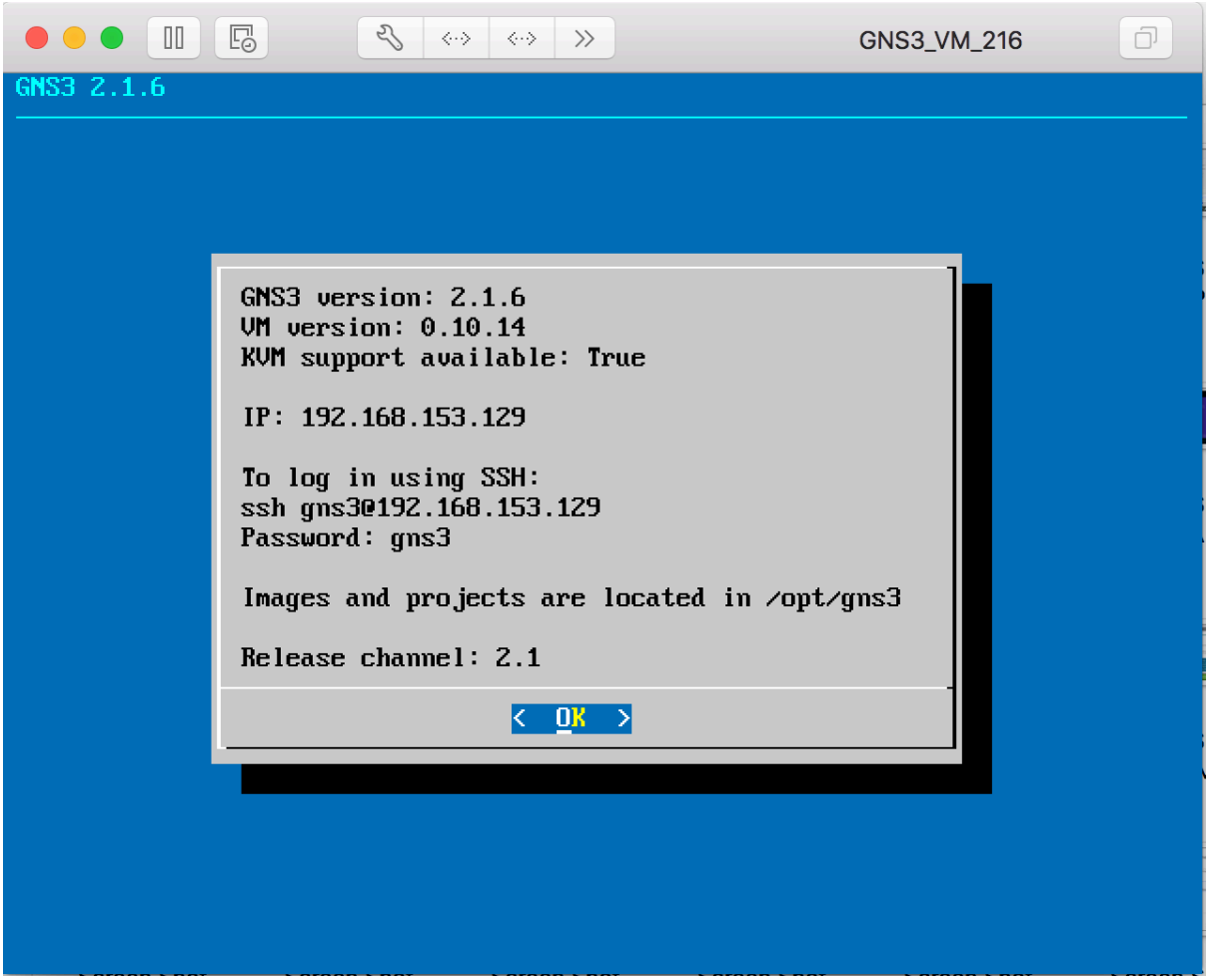
Figura 32 - Configuração da VM para o funcionando do KVM



Fonte: Autor

21˚ Passo – Após este procedimento o suporte ao modo KVM está habilitado, conforme a Figura 33.

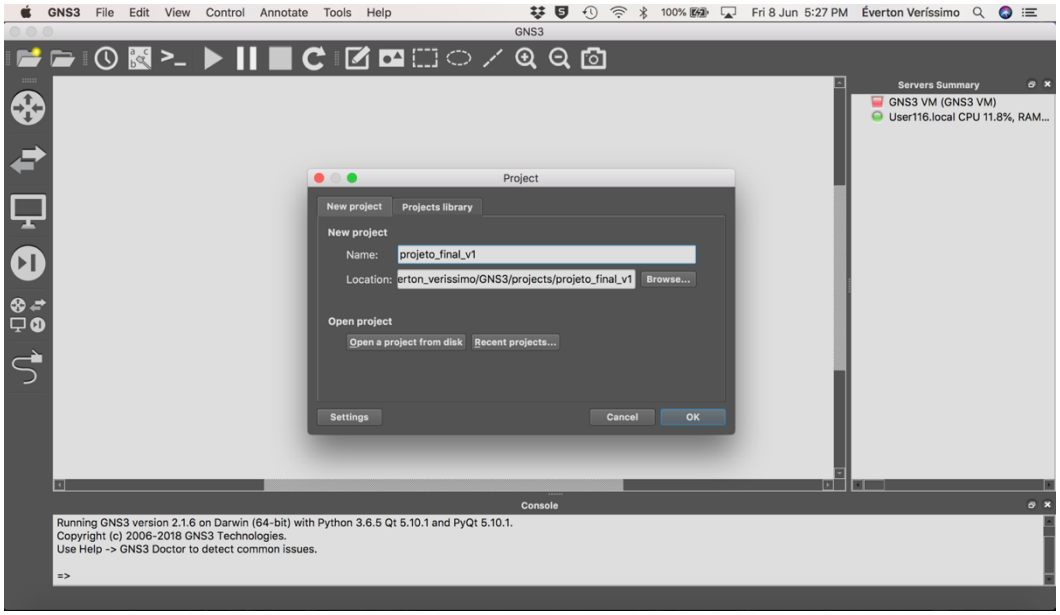
Figura 33 - GNS3 VM funcionando com suporte a KVM



Fonte: Autor

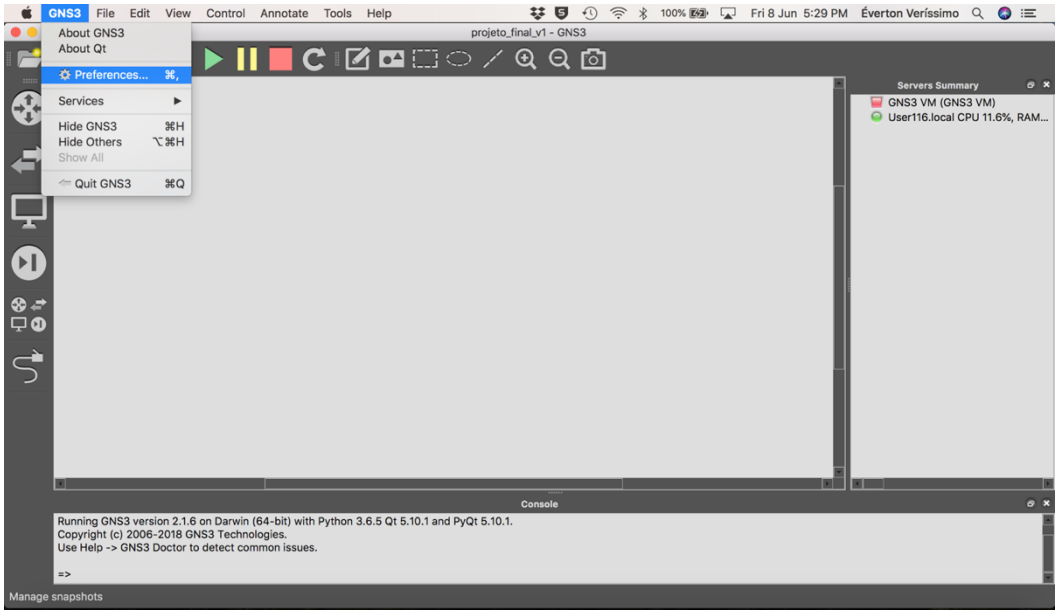
22˚ Passo – Agora pode-se integrar o GNS3.app com o GNS3 VM. Deve-se criar um projeto no GNS3.app. Parar o ambiente de emulação, clicando no quadrado vermelho e após ir em preferências, conforme as Figuras 34 e 35.

Figura 34 - Criação de um projeto no GNS3.app



Fonte: Autor

Figura 35 - Acesso a preferência do GNS3.app

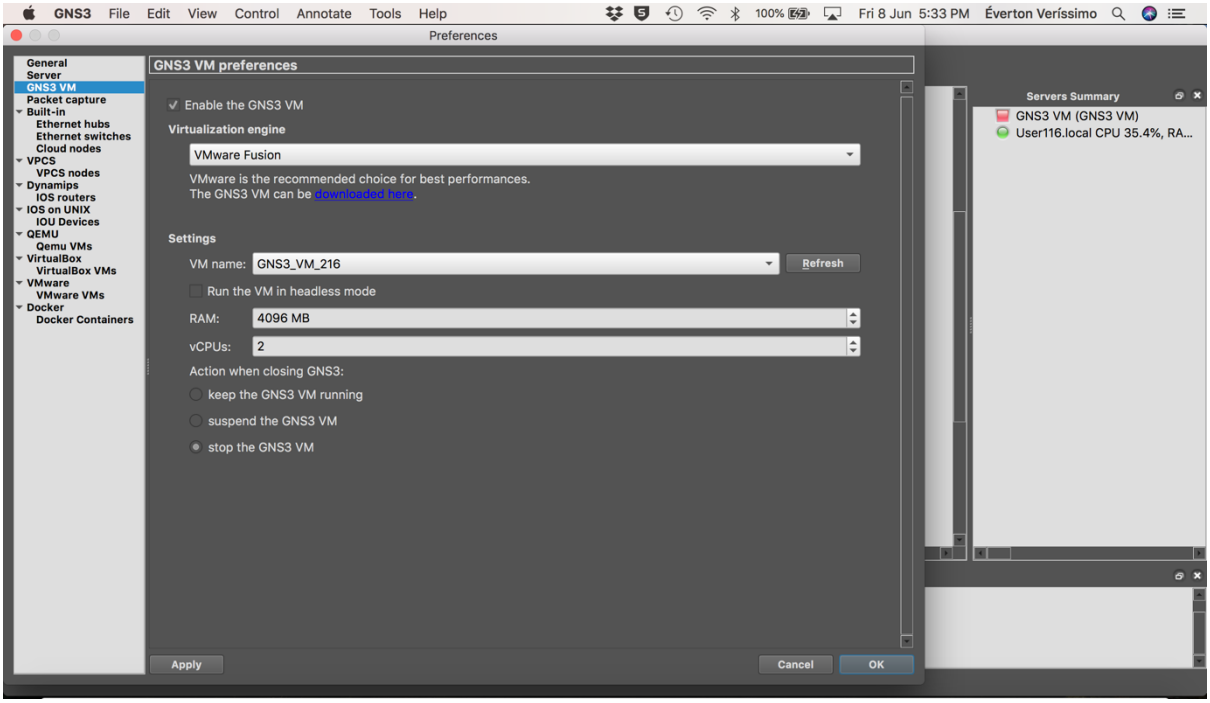


Fonte: Autor

23˚ Passo – Agora em preferências, deve-se ir em GNS3VM, habilitar a opção Enable the GNS3 VM, selecionar Virtualization Engine VMware Fusion, em VM name, deve-se

selecionar o nome da VM importada para o VMware Fusion, e por fim clicar em OK, conforme a Figura 36.

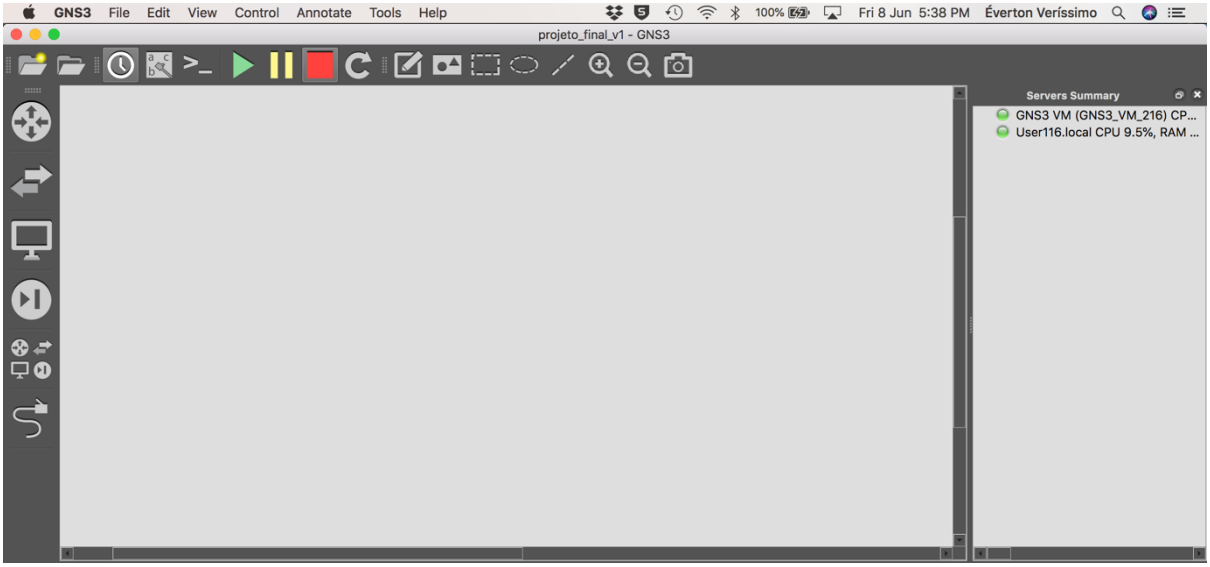
Figura 36 - Integração do GNS3.app com o GNS3 VM



Fonte: Autor

24° Passo – Para verificar se a integração foi concluída, basta visualizar na aba “servers summary”, conforme a Figura 37.

Figura 37 - Validação da integração entre o GNS3.app e GNS3 VM



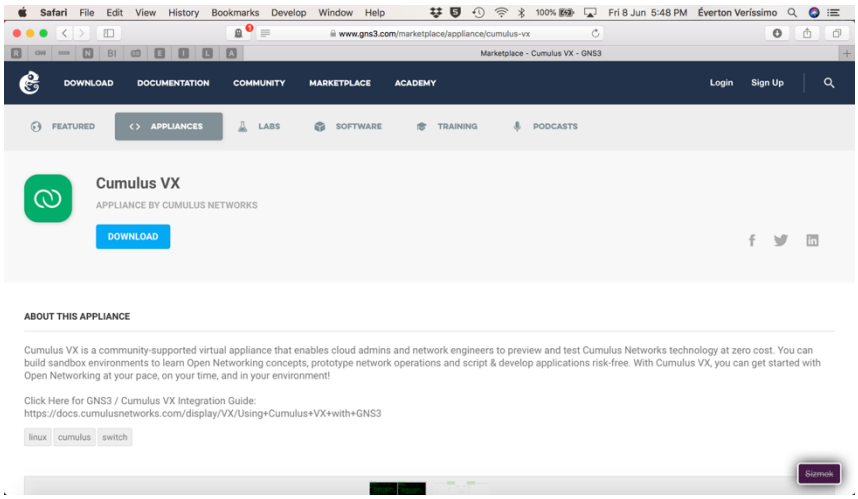
Fonte: Autor

3.3.2. INSTALAÇÃO DO SISTEMA OPERACIONAL DE REDE NO GNS3

O Cumulus VX é um sistema operacional de rede que é compatível com o GNS3. Será demostrado com é feita a instalação do Cumulus VX no ambiente de emulação no GNS3.

1° Passo – Deve-se acessar o site www.gns3.com, clicar na aba “Marketplace”, após clicar em “Appliances”, no campo de busca digitar Cumulus. É fazer o download do appliance para importação no GNS3, conforme a Figura 38.

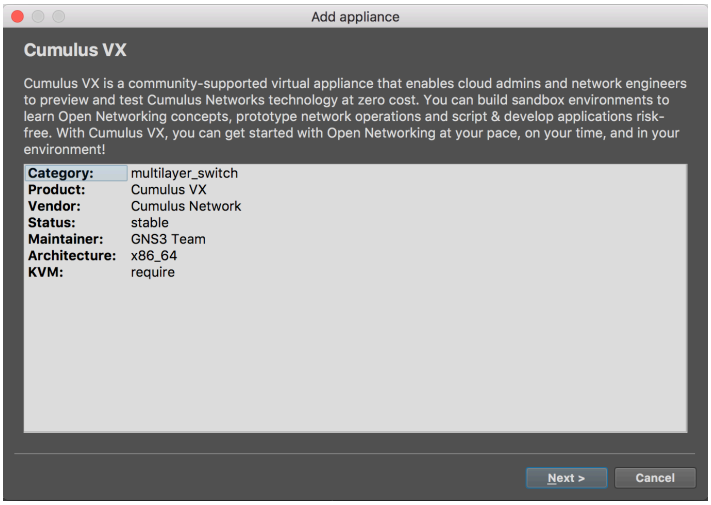
Figura 38 - Acesso ao site GNS3 para download do Appliance Cumulus



Fonte: Autor

2° Passo – Após esta etapa será baixado um arquivo com a extensão .txt que deverá ser renomeado para extensão .gns3a. Para importação deste arquivo deve-se ir em GNS3, Arquivo e importar appliance; Após importar o arquivo será aperto um assistente de importação, conforme a Figura 39.

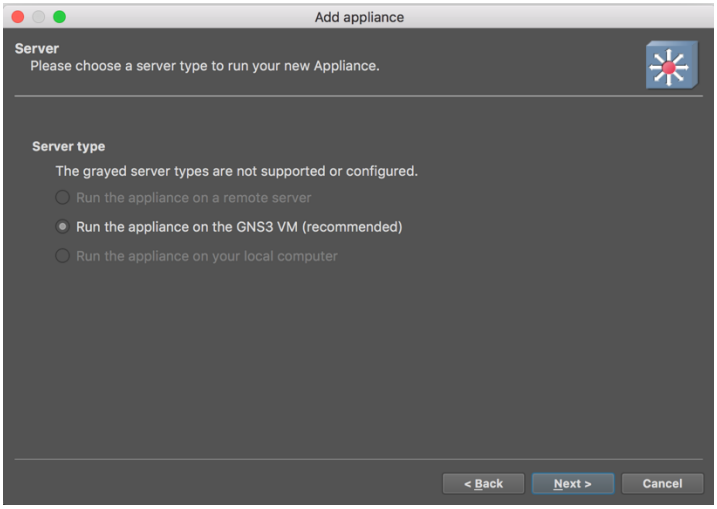
Figura 39 - Instalação do appliance Cumulus



Fonte: Autor.

3˚ Passo – Selecionar a opção “Run the appliance on the GNS3 VM”, conforme a Figura 40.

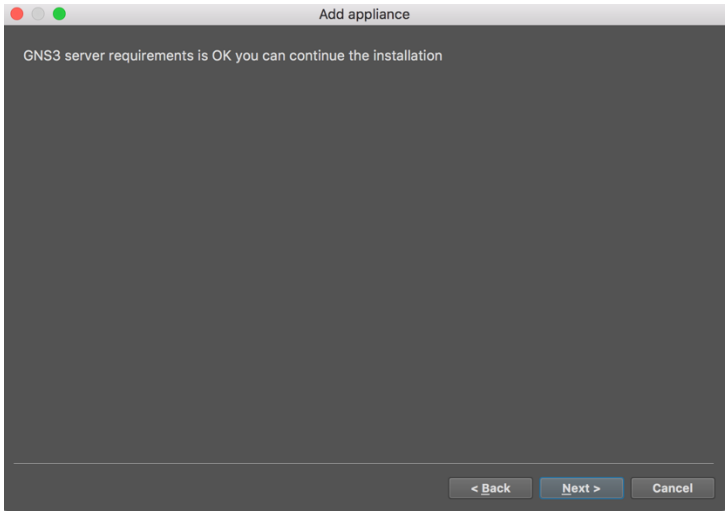
Figura 40 - Instalação do appliance do Cumulus



Fonte: Autor

4˚ Passo – Clicar em Next para prosseguir a instalação, conforme a Figura 41.

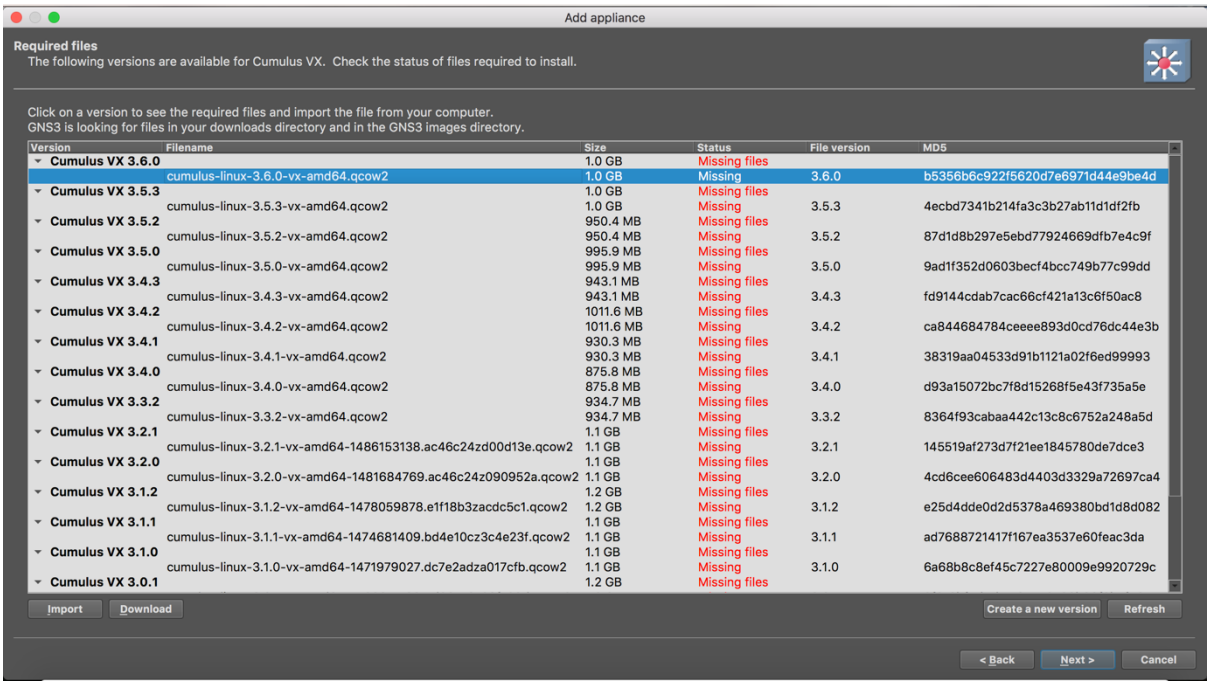
Figura 41 - Instalação do appliance do Cumulus



Fonte: Autor

5° Passo – Para prosseguir com a instalação, deve-se clicar na opção download para que o software seja baixado, conforme a Figura 42.

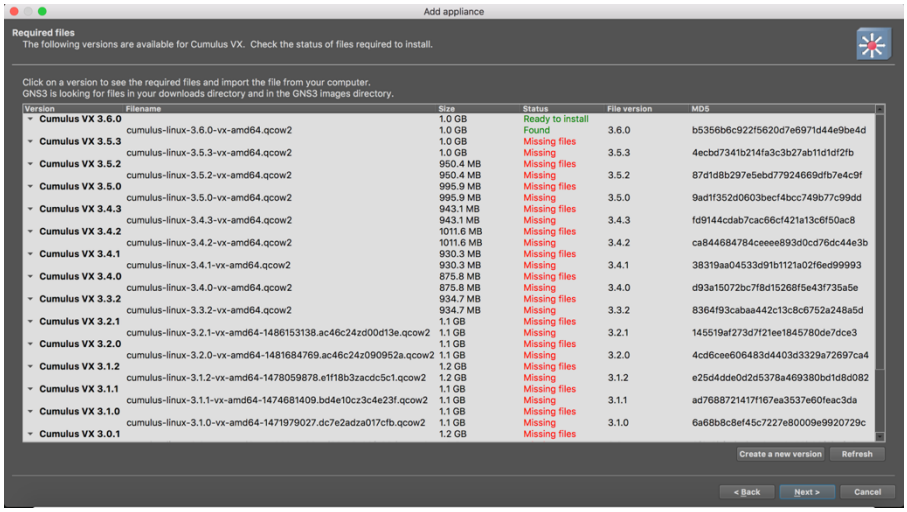
Figura 42 - Importação do Cumulus VX



Fonte: Autor

6˚ Passo – Após o download o GNS3, deve-se informar a localização do arquivo para que o GNS3 possa finalizar a importação e para prosseguir deve-se clicar em Next, conforme a Figura 43.

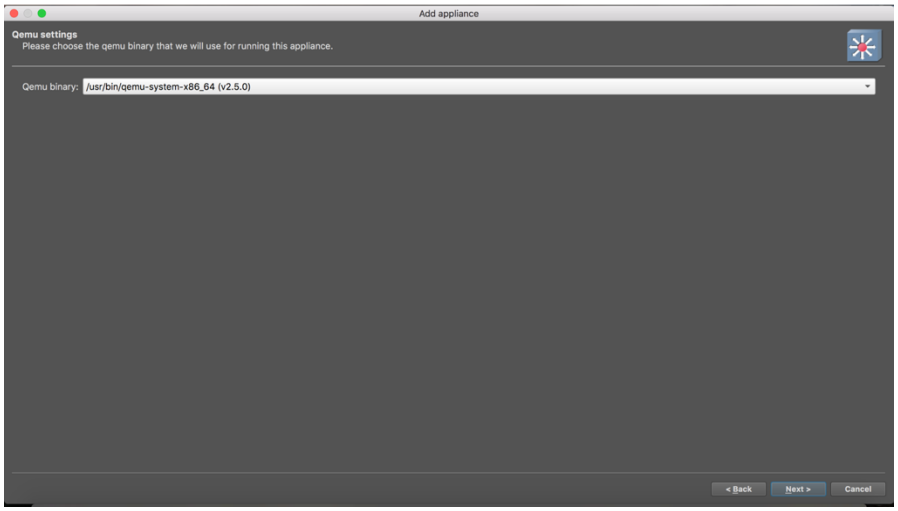
Figura 43 - Instalação do Cumulus VX



Fonte: Autor

7˚ Passo – Após deve-se selecionar a arquitetura do processador que será emulada, deve-se selecionar a opção x86_64, conforme a Figura 44.

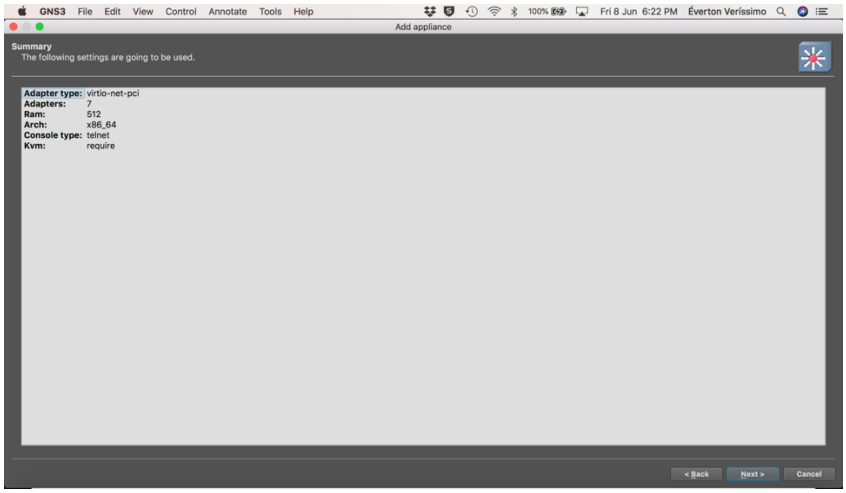
Figura 44 - Definição da arquitetura do processador para instalação



Fonte: Autor

8° Passo – Para finalizar, é mostrado uma lista com as configurações do appliance, conforme a Figura 45.

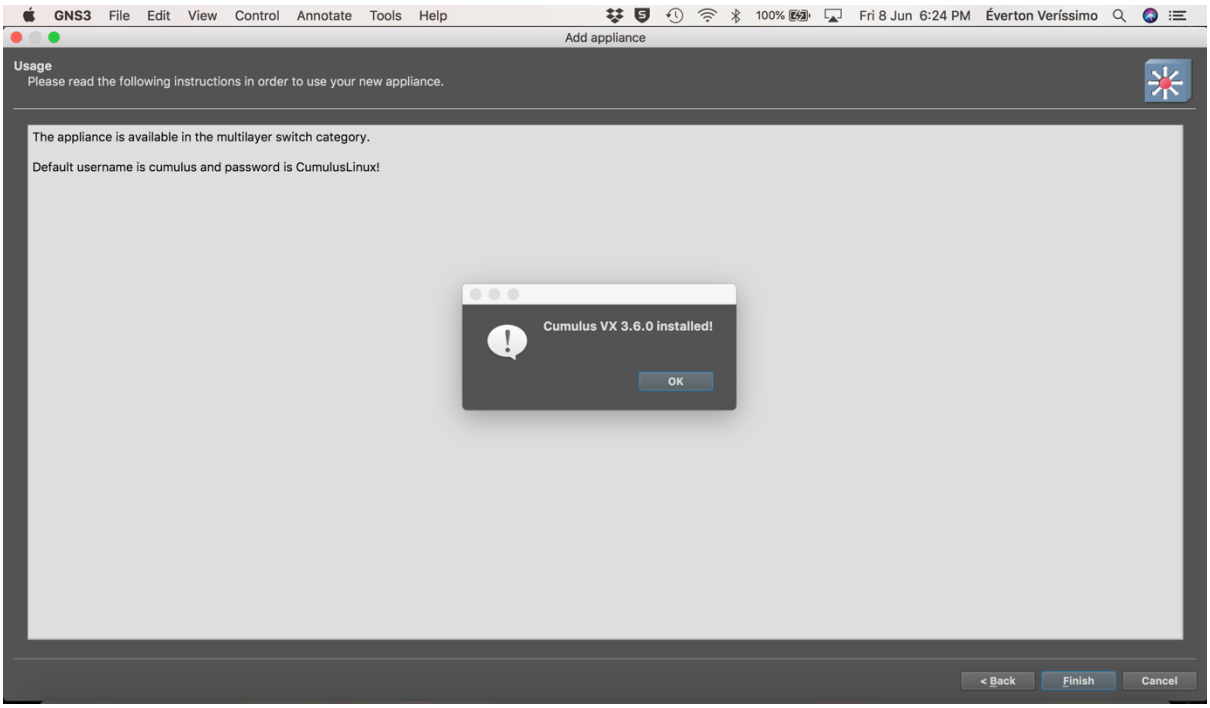
Figura 45 - Resumo das configurações para instalação do Cumulus VX



Fonte: Autor

9° Passo – Clicar em Finish é o appliance estará instalado, conforme a Figura 46.

Figura 46 - Finalização da instalação do Cumulus VX



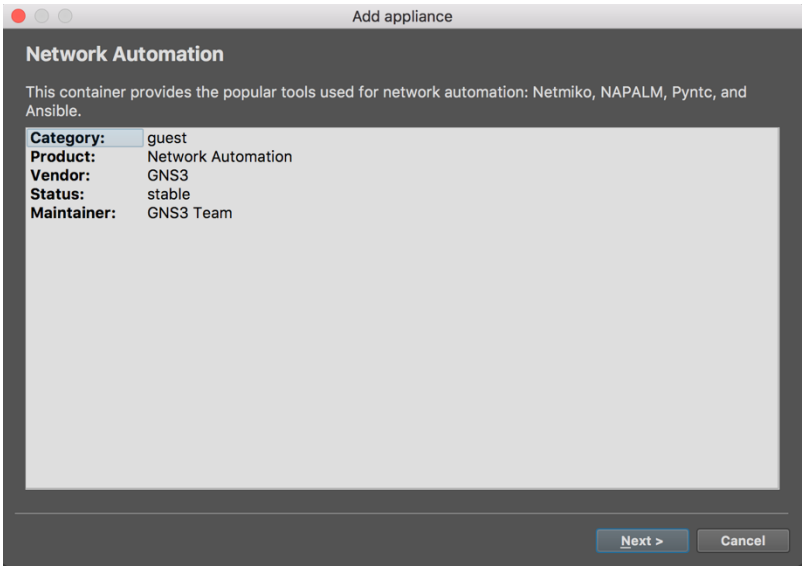
Fonte: Autor

3.3.3. INSTALAÇÃO DO SERVIDOR DE AUTOMAÇÃO NO GNS3

Para automação será utilizado um appliance que também pode ser encontrado no marketplace do GNS3, este possui o software Ansible instalado, abaixo será descrito o procedimento.

1˚ Passo – Deve-se acessar o site www.gns3.com, clicar na aba “Marketplace”, após clicar em “Appliances”, no campo de busca digitar Network Automation. Deve-se fazer o download do appliance para importação no GNS3. Será aberto um assistente de instalação, conforme a Figura 47.

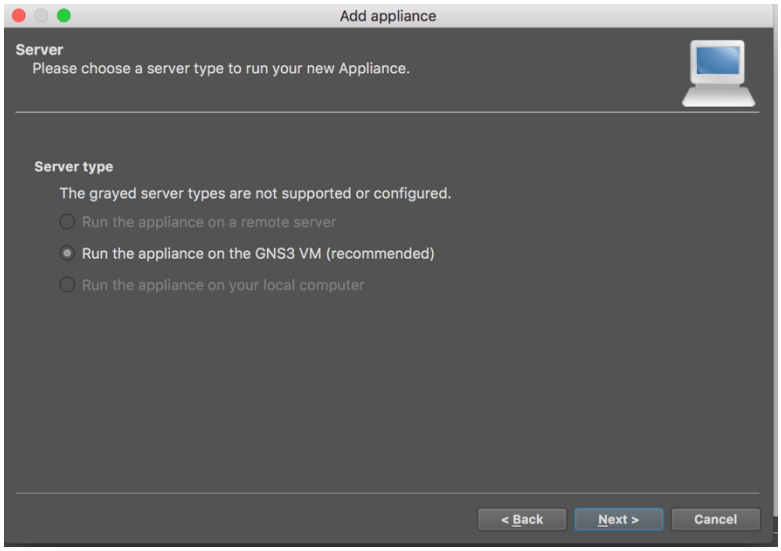
Figura 47 - Importação do appliance de automação



Fonte: Autor

2˚ Passo – Selecionar a opção “Run the appliance on the GNS3 VM”, conforme a Figura 48.

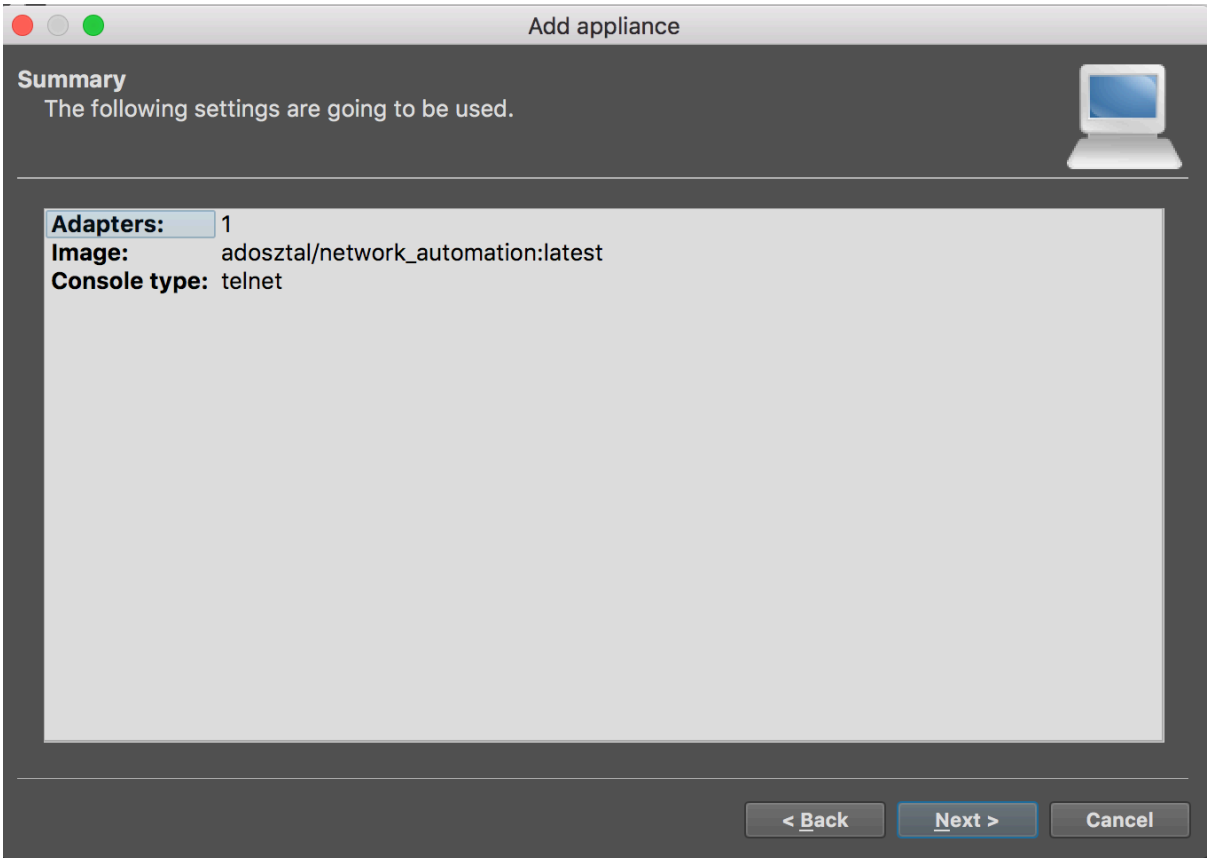
Figura 48 - Instalação do appliance de automação



Fonte: Autor

3˚ Passo – Selecionar Next, Conforme a Figura 49.

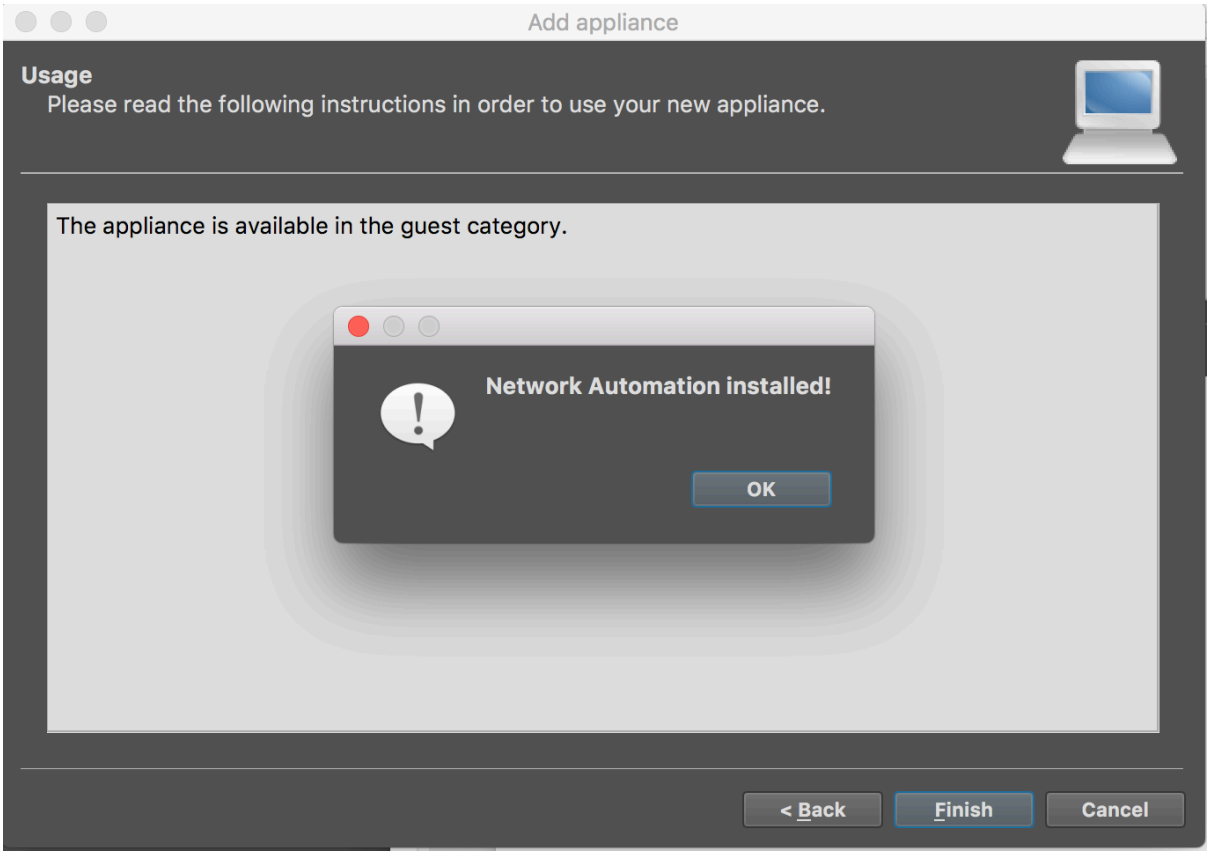
Figura 49 - Resumo das configurações para instalação do appliance de automação



Fonte: Autor

4˚ Passo – Selecionar Finish, é appliance estará instalado, conforme a Figura 50.

Figura 50 - Finalização da instalação do appliance de automação



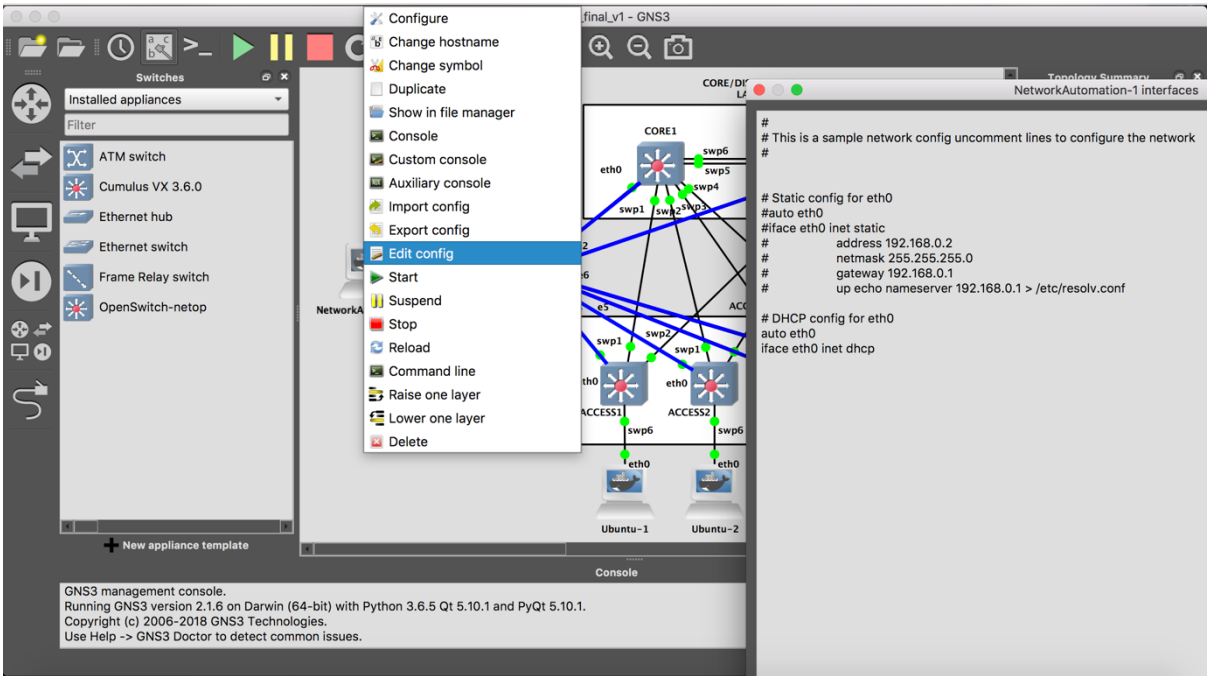
Fonte: Autor

3.3.3.1. CONFIGURAÇÃO DHCP SERVER E SERVIDOR WEB

Após a instalação do appliance servidor de automação é necessário a instalação e configuração de dois serviços. O primeiro é um DHCP Server que distribuirá os endereços para rede de gerência e o segundo será um servidor web que servirá para armazenamento do *script* de provisionamento e chaves SSH utilizadas pelo software Ansible.

1˚ Passo – Deve-se acessar a ferramenta GNS3, clicar com o botão direito, ir em Edit Config e retirar o comentário ou # antes de “auto eth0” e “iface eth0 inet dhcp” isso permitará que o servidor receba um endereço IP para baixar o DHCP server e Servidor WEB, conforme a Figura 51.

Figura 51 - Configuração do DHCP para o servidor de automação



Fonte: Autor

2˚ Passo – Verificar se o servidor tem acesso à Internet, conforme a Figura 52.

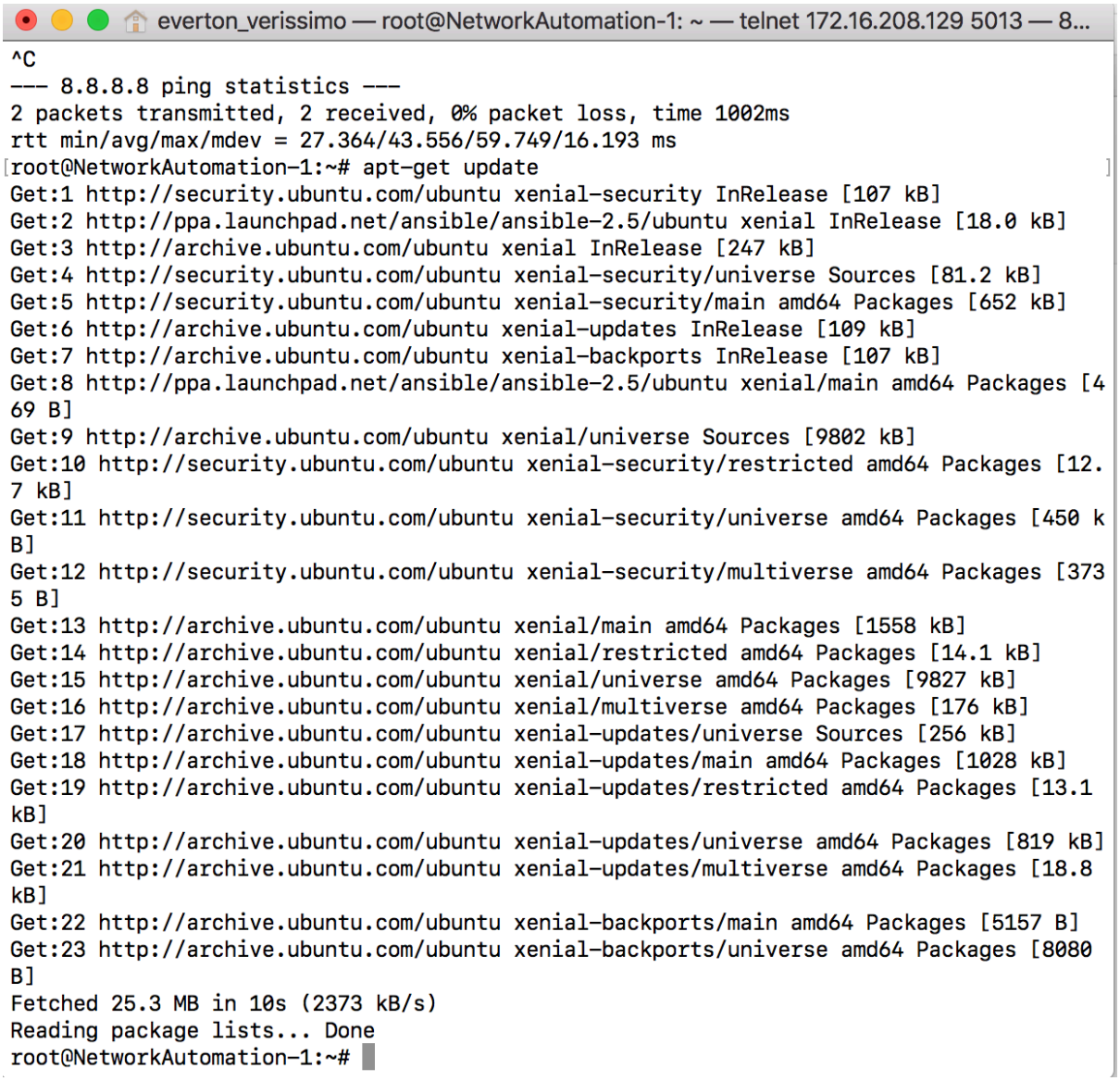
Figura 52 - Teste de conectividade

```
everton_verissimo — root@NetworkAutomation-1: ~ — telnet 172.16.208.129 5...
Connected to 172.16.208.129.
Escape character is '^]'.
NetworkAutomation-1 console is now available... Press RETURN to get started.


udhcpc (v1.24.2) started
Sending discover...
Sending discover...
Sending discover...
Sending select for 192.168.122.50...
Lease of 192.168.122.50 obtained, lease time 3600
root@NetworkAutomation-1:~#
root@NetworkAutomation-1:~#
root@NetworkAutomation-1:~#
root@NetworkAutomation-1:~# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=127 time=59.7 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=127 time=27.3 ms
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 27.364/43.556/59.749/16.193 ms
root@NetworkAutomation-1:~#
```

Fonte: Autor

3˚ Passo – Atualizar os pacotes utilizando o comando apt-get update, conforme a Figura 53.

Figura 53 - Atualização de pacotes



Fonte: Autor

4˚ Passo – Instalar o servidor web utilizando o comando apt-get install apache2, após iniciar a instalação aparecerá uma mensagem perguntando se quer instalar o software. Pressione Y e enter para prosseguir a instalação, conforme a Figura 54.

Figura 54 - Instalação servidor web

```
root@NetworkAutomation-1:~# apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libicu55 liblua5.1-0 libxml2
  sgml-base ssl-cert xml-core
Suggested packages:
  www-browser apache2-doc apache2-suexec-pristine | apache2-suexec-custom ufw
  sgml-base-doc openssl-blacklist debhelper
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libicu55 liblua5.1-0 libxml2
  sgml-base ssl-cert xml-core
0 upgraded, 14 newly installed, 0 to remove and 14 not upgraded.
Need to get 9937 kB of archives.
After this operation, 39.7 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Fonte: Autor

5˚ Passo – Instalar o DHCP Server utilizando o comando apt-get install isc-dhcp-server, após iniciar a instalação aparecerá uma mensagem perguntando se quer instalar o software. Pressione Y e enter para prosseguir a instalação, conforme a Figura 55.

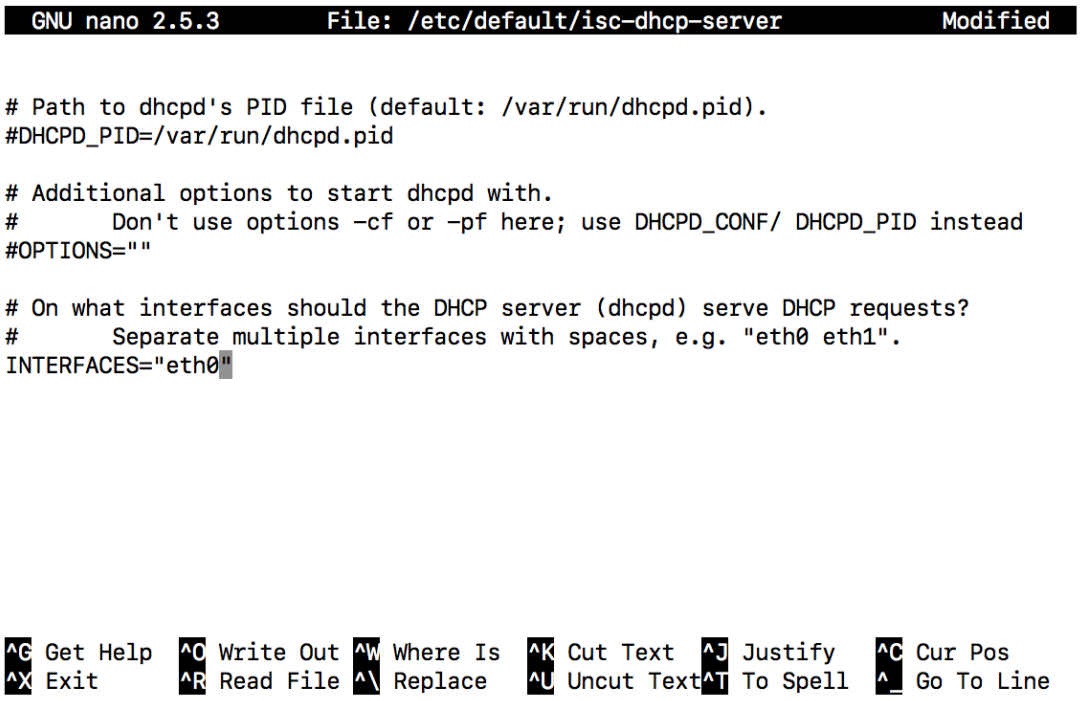
Figura 55 - Instalação DHCP Server

```
everton_verissimo — root@NetworkAutomation-1: ~ — telnet 172.16.208.129 5013 — 92×45
Reading package lists... Done
root@NetworkAutomation-1:~# apt-get install isc-dhcp-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  isc-dhcp-common libdns-export162 libirs-export141 libisc-export160
  libisccfg-export140
Suggested packages:
  isc-dhcp-server-ldap apparmor policycoreutils
The following NEW packages will be installed:
  isc-dhcp-common isc-dhcp-server libdns-export162 libirs-export141
  libisc-export160 libisccfg-export140
0 upgraded, 6 newly installed, 0 to remove and 14 not upgraded.
Need to get 1394 kB of archives.
After this operation, 4163 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 isc-dhcp-common amd64 4.3.3
-5ubuntu12.10 [105 kB]
Get:2 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 libisc-export160 amd64 1:9.
10.3.dfsg.P4-8ubuntu1.10 [153 kB]
Get:3 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 libdns-export162 amd64 1:9.
10.3.dfsg.P4-8ubuntu1.10 [666 kB]
Get:4 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 libisccfg-export140 amd64 1
:9.10.3.dfsg.P4-8ubuntu1.10 [38.5 kB]
Get:5 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 libirs-export141 amd64 1:9.
10.3.dfsg.P4-8ubuntu1.10 [17.5 kB]
Get:6 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 isc-dhcp-server amd64 4.3.3
-5ubuntu12.10 [414 kB]
Fetched 1394 kB in 2s (465 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package isc-dhcp-common.
(Reading database ... 17048 files and directories currently installed.)
Preparing to unpack .../isc-dhcp-common_4.3.3-5ubuntu12.10_amd64.deb ...
Unpacking isc-dhcp-common (4.3.3-5ubuntu12.10) ...
Selecting previously unselected package libisc-export160.
Preparing to unpack .../libisc-export160_1%3a9.10.3.dfsg.P4-8ubuntu1.10_amd64.deb ...
Unpacking libisc-export160 (1:9.10.3.dfsg.P4-8ubuntu1.10) ...
Selecting previously unselected package libdns-export162.
Preparing to unpack .../libdns-export162_1%3a9.10.3.dfsg.P4-8ubuntu1.10_amd64.deb ...
Unpacking libdns-export162 (1:9.10.3.dfsg.P4-8ubuntu1.10) ...
Selecting previously unselected package libisccfg-export140.
Preparing to unpack .../libisccfg-export140_1%3a9.10.3.dfsg.P4-8ubuntu1.10_amd64.deb ...
Unpacking libisccfg-export140 (1:9.10.3.dfsg.P4-8ubuntu1.10) ...
Selecting previously unselected package libirs-export141.
```

Fonte: Autor

6˚ Passo – Deve-se acessar o arquivo isc-dhcp-server através do comando nano /etc/default/isc-dhcp-server e incluir a informação INTERFACES=“eth0”. Pressionar Ctrl + Z, Y para salvar e Enter, conforme a Figura 56.

Figura 56 - Configuração do arquivo isc-dhcp-server



Fonte: Autor

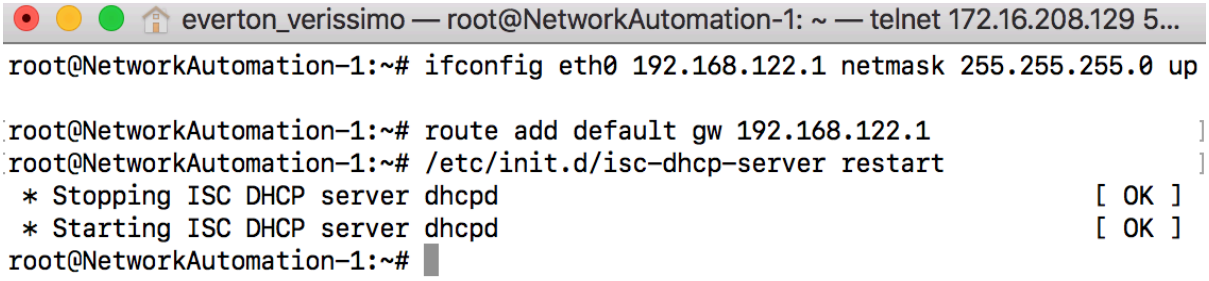
7˚ Passo – Agora deve-se acessar o arquivo dhcpd.conf através do comando nano /etc/dhcp/dhcpd.conf e incluir as informações abaixo. Pressionar Ctrl + Z, Y para salvar e enter.

```
option domain-name "teste.lab";
option domain-name-servers NetworkAutomation-1.teste.lab;

authoritative;

option cumulus-provision-url code 239 = text;

subnet 192.168.122.0 netmask 255.255.255.0 {
  range 192.168.122.10 192.168.122.20;
  option subnet-mask 255.255.255.0;
  option routers 192.168.122.1;
  option broadcast-address 192.168.122.255;
  option cumulus-provision-url "http://192.168.122.1/demo.sh";
}
```

```
host core1 {
        hardware ethernet 0c:fc:4a:68:a1:00;
        fixed-address 192.168.122.10;
        option host-name "core1";
}

host core2 {
        hardware ethernet 0c:fc:4a:d3:b9:00;
        fixed-address 192.168.122.11;
        option host-name "core2";
}

host acesso1 {
        hardware ethernet 0c:fc:4a:4c:d1:00;
        fixed-address 192.168.122.12;
        option host-name "acesso1";
}

host acesso2 {
        hardware ethernet 0c:fc:4a:42:6f:00;
        fixed-address 192.168.122.13;
        option host-name "acesso2";
}

host acesso3 {
        hardware ethernet 0c:fc:4a:d7:c9:00;
        fixed-address 192.168.122.14;
        option host-name "acesso3";
}

host acesso4 {
        hardware ethernet 0c:fc:4a:5f:12:00;
        fixed-address 192.168.122.15;
        option host-name "acesso4";
}
```

8˚ Passo – Este passo deve-se trocar o endereço da interface eth0 do servidor de automação para que ele passe a ser o servidor DHCP na rede. Para isso deve-se usar os comando ifconfig eth0 192.168.122.1 netmask 255.255.255.0 up e route add default gw 192.168.122.1. Após isso deve-se reiniciar o serviço através do comando /etc/init.d/isc-dhcp-server restart, conforme a Figura 57.

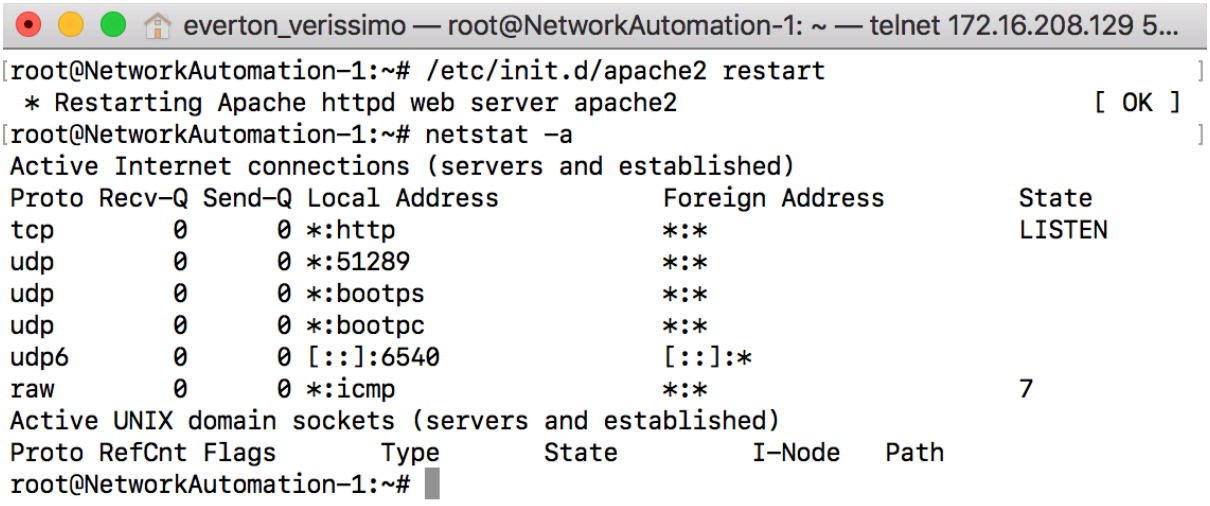
Figura 57 - Configuração IP e reinício do DHCP Server



Fonte: Autor

9° Passo – Deve-se acessar o arquivo apache2.conf através do comando nano /etc/apache/apache2.conf e incluir no arquivo o linha ServerName NetworkAutomation-1 e reiniciar o serviço através do comando /etc/init.d/apache2 restart . Pode-se verificar o funcionamento do servidor web utilizando o comando netstat -a, conforme a Figura 58.

Figura 58 - Reinício servidor web e verificação porta http

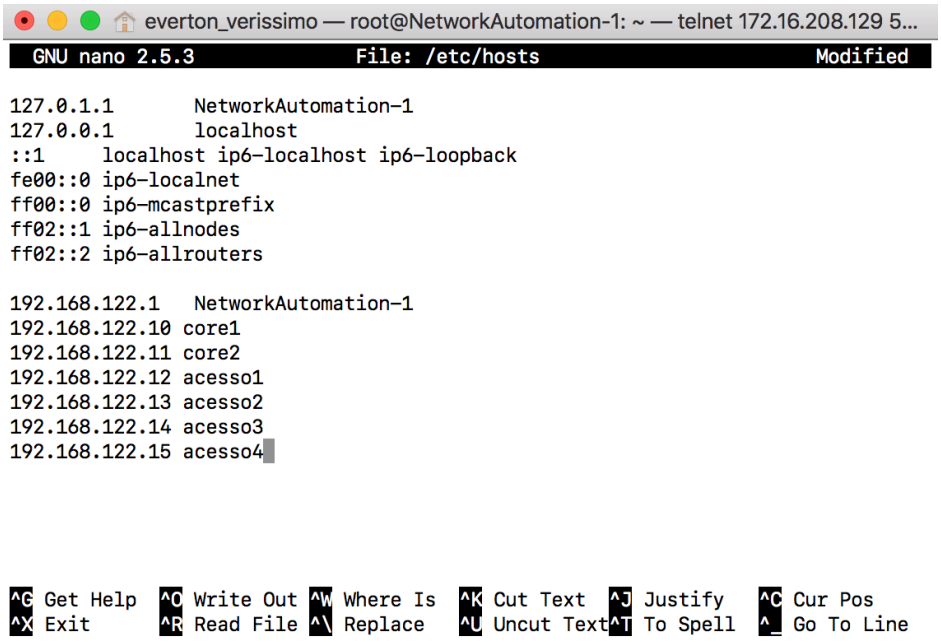


Fonte: Autor

3.3.3.2. INSTALAÇÃO E CONFIGURAÇÃO ANSIBLE

1° Passo – O Ansible vem instalado por padrão no Appliance de Automação. A sua instalação também pode ser feita através dos comandos: apt-get install software-properties-common; apt-add-repository ppa:ansible/ansible; apt-get update; apt-get install ansible. Para a configuração do Ansible precisa-se alterar o arquivo hosts do servidor de automação através do commando nano /etc/hosts vamos incluir os nomes e endereços IPs, conforme a Figura 59.

Figura 59 - Configuração do arquivo hosts no servidor de automação



Fonte: Autor

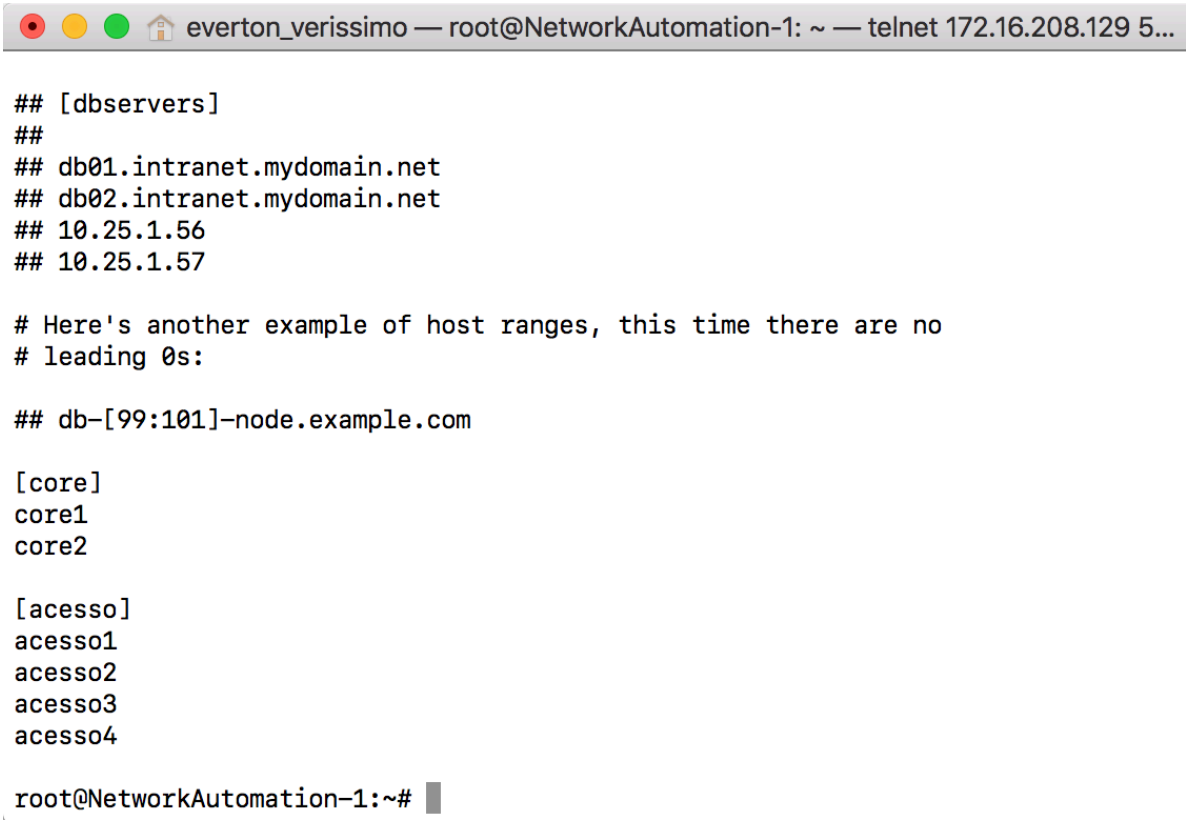
2° Passo – Deve-se acessar o arquivo ansible.cfg através do comando nano /etc/ansible/ansible.cfg e alterar o parâmetro inventory, conforme a Figura 60.

Figura 60 - Configuração arquivo ansible.cfg

Fonte: Autor

3° Passo – Deve-se acessar o arquivo hosts do ansible através do comando nano /etc/ansible/hosts e incluir o nome dos switches separados em dois grupos [core] com core1 core2 e [acesso] com acesso1, acesso2, acesso3, acesso4, conforme a Figura 61.

Figura 61- Configuração do arquivo /etc/ansible/hosts



Fonte: Autor

CAPÍTULO 4 - IMPLEMENTAÇÃO E RESULTADOS

Neste capítulo do trabalho será apresentado o descritivo do ambiente de emulação, demonstração da solução, resultado dos testes e avaliação da proposta de solução para automação e provisionamento de uma rede campus. Também será detalhado alguns dificultados encontrados durante os testes.

4. PROVISIONAMENTO

Basicamente temos dois métodos de provisionamentos nos switches open networking, o ONIE cria um ambiente possibilitando a instalação de um sistema operacional de rede em um bare metal ou quando utilizando um white box switch ou um brite box switch o ONIE cria um ambiente onde é possível a alteração do sistema operacional de rede. A outra forma é o ZTP que faz um provisionamento de configurações.

Para o ambiente de teste serão demonstração aos dois métodos. Para isso será utilizado as configurações previamente estabelecidas no DHCP Server e no servidor web. No próximo item serão o provisionamento utilizando o ONIE e o ZTP.

4.1.1. ONIE

Para realização dos testes com o ONIE foram utilizados o DHCP Server e o Servidor web instalados no servidor de automação, após o download da nova versão do sistema operacional deve-se adicionar este arquivo no servidor web. Deve-se então acessar o switch que será atualizado é executa um comando. Os procedimentos dos testes serão detalhados abaixo.

1˚ Passo - Deve-se baixar o novo sistema operacional de rede. Para isso é necessário baixar a ferramenta wget através do comando apt-get install wget, após este comando, deve-se efetuar o download através do comando wget -c -P /var/www/html https://s3.amazonaws.com/cumulusfiles/CumulusLinux-3.6.1/cumulus-linux-3.6.1-vx-amd64.bin . A opção -c permite que o download seja reestabelecidocaso o download seja interrompido e a opção -P permite salvar o download em um caminho específico, conforme Figura 62.

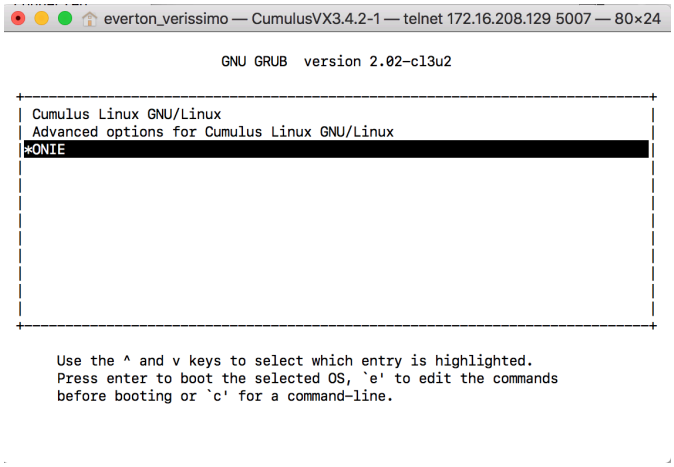
Figura 62 - Download software wget e sistema operacional de rede

```
everton_verissimo — root@NetworkAutomation-1: ~ — telnet 172.16.208.129 5027 — 89×...
root@NetworkAutomation-1:~# apt-get install wget
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  wget
0 upgraded, 1 newly installed, 0 to remove and 14 not upgraded.
Need to get 299 kB of archives.
After this operation, 905 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 wget amd64 1.17.1-1ubunt
u1.4 [299 kB]
Fetched 299 kB in 1s (198 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package wget.
(Reading database ... 17891 files and directories currently installed.)
Preparing to unpack .../wget_1.17.1-1ubuntu1.4_amd64.deb ...
Unpacking wget (1.17.1-1ubuntu1.4) ...
Setting up wget (1.17.1-1ubuntu1.4) ...
root@NetworkAutomation-1:~# wget -c -P /var/www/html https://s3.amazonaws.com/cumulusfile
s/CumulusLinux-3.6.1/cumulus-linux-3.6.1-vx-amd64.bin
--2018-06-16 02:07:59--  https://s3.amazonaws.com/cumulusfiles/CumulusLinux-3.6.1/cumulus
-linux-3.6.1-vx-amd64.bin
Resolving s3.amazonaws.com (s3.amazonaws.com)... 52.216.129.141
Connecting to s3.amazonaws.com (s3.amazonaws.com)|52.216.129.141|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 165631231 (158M) [application/octet-stream]
Saving to: '/var/www/html/cumulus-linux-3.6.1-vx-amd64.bin'

cumulus-linux-3.6.1 100%[===================>] 157.96M   667KB/s    in 49s

2018-06-16 02:08:49 (3.21 MB/s) - '/var/www/html/cumulus-linux-3.6.1-vx-amd64.bin' saved
[165631231/165631231]
```

Fonte: Autor

2˚ Passo – Após o passo anterior, deve-se ligar o equipamento é selecionar a opção ONIE, conforme a Figura 63.

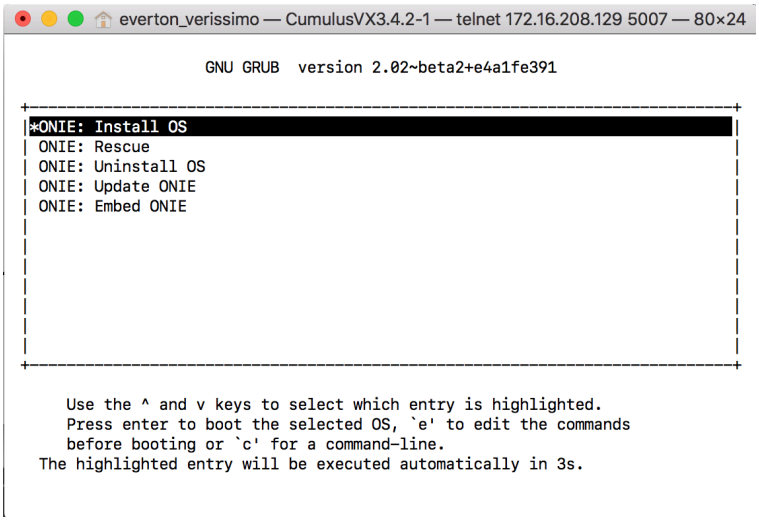
Figura 63 - Inicialização ONIE



Fonte: Autor

3˚ Passo – Após ONIE: Install OS, conforme a Figura 64.

Figura 64 - Opção de instalação no ONIE



Fonte: Autor

4˚ Passo – Na console do ONIE, deve-se executar o comando onie-nos-install http://192.168.122.1/onie-installer-cumulus-linux-3.6.1-vx-amd64.bin, conforme a Figura 64.

Figura 65 - Download e instalação do sistema operacional de rede através do ONIE

```
everton_verissimo — CumulusVX3.4.2-1 — telnet 172.16.208.129 5007 — 80×24
[ONIE:/ #                                                                  ]
[ONIE:/ #                                                                  ]
[ONIE:/ #                                                                  ]
[ONIE:/ #                                                                  ]
[ONIE:/ #                                                                  ]
[ONIE:/ #                                                                  ]
[ONIE:/ #                                                                  ]
[ONIE:/ #                                                                  ]
[ONIE:/ #                                                                  ]
ONIE:/ # onie-nos-install http://192.168.122.1/onie-installer-cumulus-linux-3.6.
[1-vx-amd64.bin                                                            ]
discover: installer mode detected.
Stopping: discover... done.
Info: Fetching http://192.168.122.1/onie-installer-cumulus-linux-3.6.1-vx-amd64.
bin ...
Connecting to 192.168.122.1 (192.168.122.1:80)
installer            100% |********************************|   157M  0:00:00 ETA
ONIE: Executing installer: http://192.168.122.1/onie-installer-cumulus-linux-3.6
.1-vx-amd64.bin
Verifying image checksum ...OK.
Preparing image archive ... OK.
Verifying image compatibility ...OK.
Verifying system ram ...OK.
```

Fonte: Autor

5˚ Passo – Após este passo o equipamento será reiniciado, após o reinicio, o sistema operacional de rede estará atualizado, conforme a Figura 66.

Figura 66 - Sistema operacial atualizado através do ONIE

```
Debian GNU/Linux 8 cumulus ttyS0

cumulus login: cumulus
Password:
Last login: Sat Jun 16 03:16:43 UTC 2018 on ttyS0
Linux cumulus 4.1.0-cl-7-amd64 #1 SMP Debian 4.1.33-1+cl3u13 (2018-04-25) x86_64

Welcome to Cumulus VX (TM)

Cumulus VX (TM) is a community supported virtual appliance designed for
experiencing, testing and prototyping Cumulus Networks' latest technology.
For any questions or technical support, visit our community site at:
http://community.cumulusnetworks.com

The registered trademark Linux (R) is used pursuant to a sublicense from LMI,
the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide
basis.
cumulus@cumulus:~$ net show system
Cumulus VX
Cumulus Linux 3.6.1
Build: Cumulus Linux 3.6.1
Uptime: 0:01:14.210000
cumulus@cumulus:~$
```

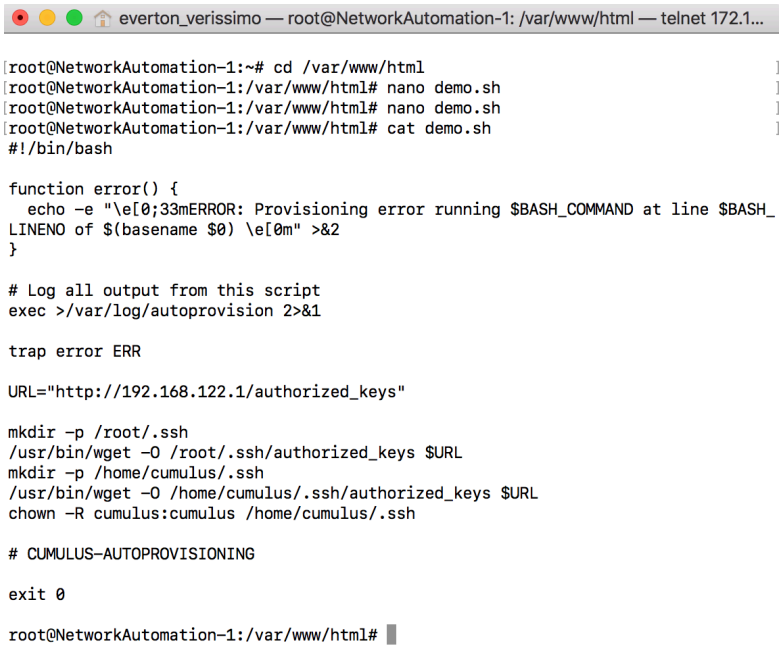
Fonte: Autor

4.1.2. ZERO TOUCH PROVISIONING - ZTP

Para realização dos testes com o ZTP foram utilizados o DHCP Server e o Servidor web instalados no servidor de automação, um script de provisionamento foi criado e salvo no servidor web. Os equipamentos receberam a informação de onde foi armazenado o *script* de provisionamento através do campo options 239 do DHCP. Todo procedimento poderá ser executado de forma automática e sem intervenção humana. O procedimento abaixo demostra os resultados do teste.

1˚ Passo – Utilizando o servidor de automação deve-se criar o script de provisionamento é salvar no servidor web. Para isso foi executado o comando cd /var/www/html e nano demo.sh para criar o script de provisionamento. Pressionar Ctrl + Z, Y para salvar e Enter, conforme a Figura 67.

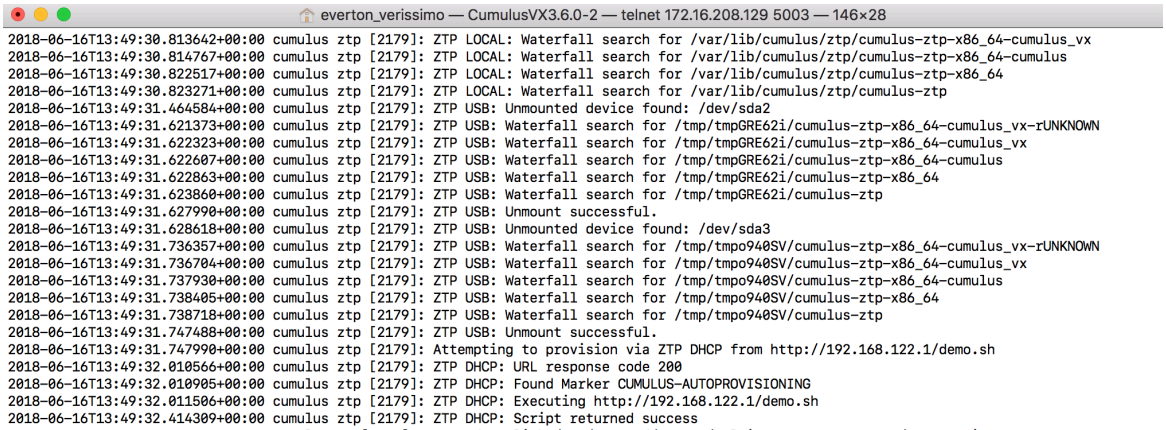
Figura 67 - Criação do script de provisionamento



Fonte: Autor

2˚ Passo – Assim que o equipamento for iniciado, ele executará o script de provisionamento. Para validar ser o provisionamento foi executado pode-se utilizar o comando sudo cat /var/log/syslog | grep ztp, conforme a Figura 68.

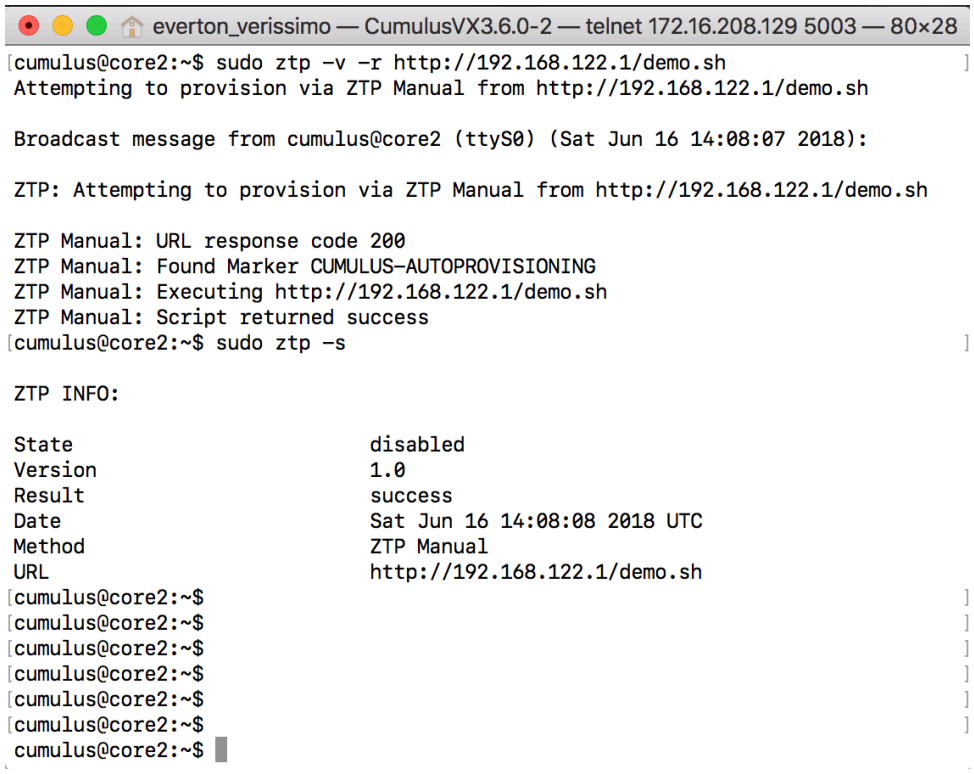
Figura 68 - Provisionamento através de DHCP



Fonte: Autor

3˚ Passo – Também pode-se forçar o provisionamento através do comando sudo ztp -v -r http://192.168.122.1/demo.sh e utilizar o comando sudo ztp -s para validar o provisionamento, conforme a Figura 69.

Figura 69 - Provisionamento manual



Fonte: Autor

4.2. AUTOMAÇÃO

No item seguir serão demostrados como criou-se os *scripts* de configuração das funções de switching, routing, troubleshooting, criação de backups, restauração de backups e uma avaliação de desempenho.

4.2.1. SWITCHING

1˚ Passo – O Cumulus por *default* não vem com as interfaces de rede habilitadas. O playbook switching.yml cria as interfaces de rede, conforme a Figura 70.

```
---
- hosts: all
  become: yes
  remote_user: cumulus
  gather_facts: no
  vars:
    ansible_user: "cumulus"
    ansible_ssh_pass: "CumulusLinux!"
    ansible_become_pass: "CumulusLinux!"
```

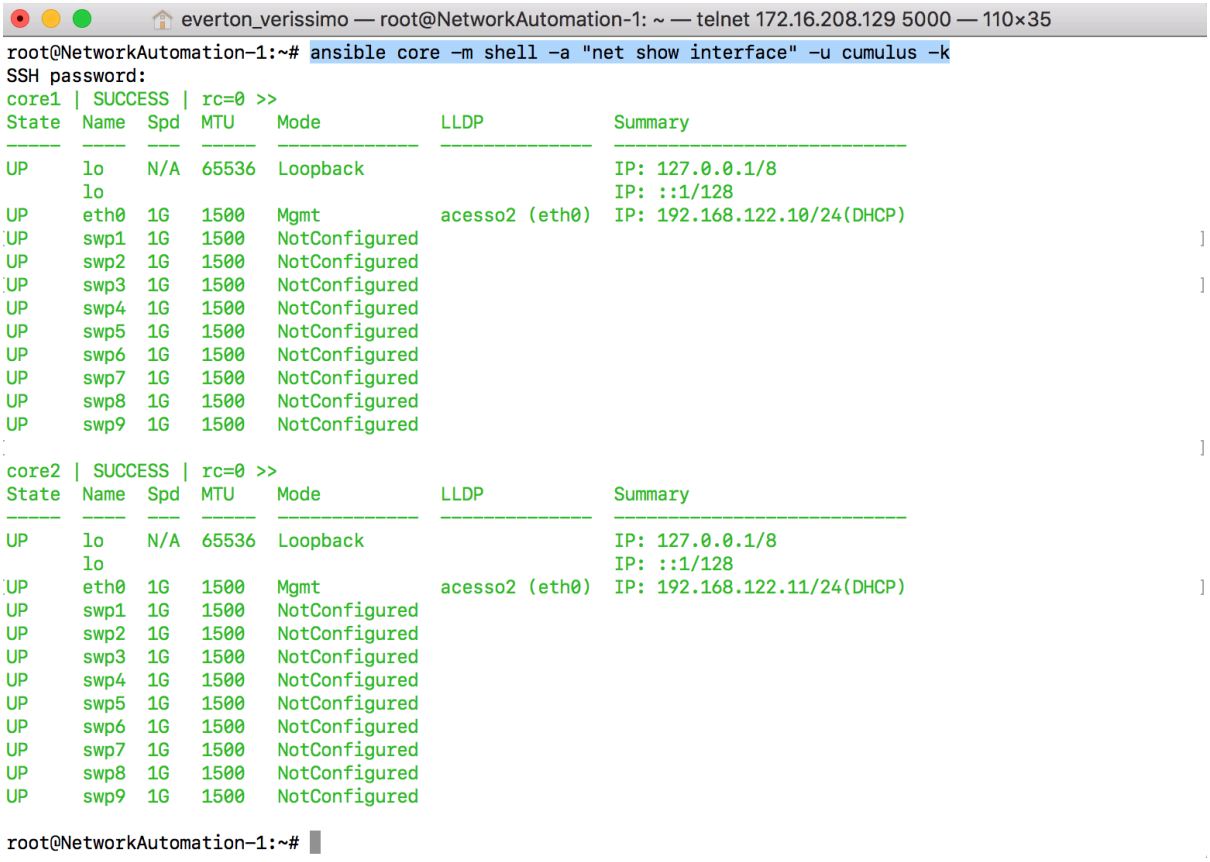
```
tasks:
  - name: adiciona as interfaces swp1 a swp9
    nclu:
      commands:
        - add int swp1
        - add int swp2
        - add int swp3
        - add int swp4
        - add int swp5
        - add int swp6
        - add int swp7
        - add int swp8
        - add int swp9


  - name: commit the changes
    nclu:
      commit: yes
```

Figura 70 - Execução do playbook switching.yml

Fonte: Autor

2˚ Passo – Pode-se validar as configurações com o commando ansible core -m shell -a "net show interface" -u cumulus -k. Conforme a Figura 71.

Figura 71 - Validação das configuções de switching



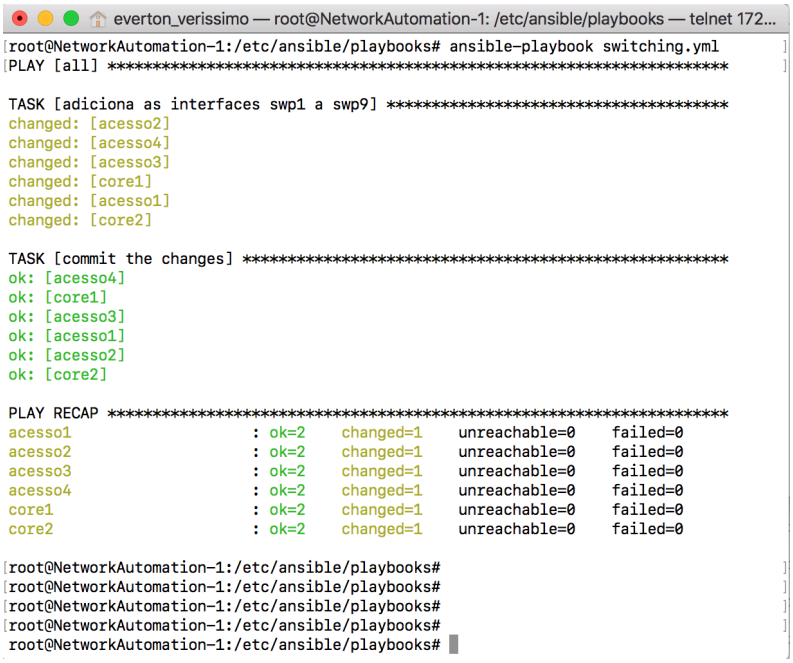
Fonte: Autor

3° Passo – Pode-se também criar um playbook para criação de VLAN e atribuição as portas do switche. O playbook vlan.yml cria as VLAN 10, 20, 30 e 40 e as associa nas portas de 1 à 8, conforme a Figura 72

```
---
- hosts: all
  become: yes
  remote_user: cumulus
  gather_facts: no
  vars:
    ansible_user: "cumulus"
    ansible_ssh_pass: "CumulusLinux!"
    ansible_become_pass: "CumulusLinux!"

  tasks:
    - name: adiciona vlans  swp1 a swp8
      nclu:
```

```
    commands:
      - add vlan 10
      - add vlan 20
      - add vlan 30
      - add vlan 40
      - add int swp1 bridge access 10
      - add int swp2 bridge access 10
      - add int swp3 bridge access 20
      - add int swp4 bridge access 20
      - add int swp5 bridge access 30
      - add int swp6 bridge access 30
      - add int swp7 bridge access 40
      - add int swp8 bridge access 40

- name: commit the changes
  nclu:
    commit: yes
```

Figura 72 - Criação de VLANs



Fonte: Autor

4˚ Passo – Pode-se também definir os endereços ips que devem ser atribuídos as respectivas interfaces. O playbook ip_interfaces.yml as interfaces IP para respectivas VLANs 10, 20, 30 e 40, conforme a Figura 73.

```
---
- hosts: core1
  become: yes
  remote_user: cumulus
  gather_facts: no
  vars:
    ansible_user: "cumulus"
    ansible_ssh_pass: "CumulusLinux!"
    ansible_become_pass: "CumulusLinux!"

  tasks:
    - name: adiciona ip na VLAN 1 CORE1
      nclu:
        commands:
          - add vlan 10 ip address 10.0.0.1/24
          - add vlan 20 ip address 20.0.0.1/24
          - add vlan 30 ip address 30.0.0.1/24
          - add vlan 40 ip address 40.0.0.1/24

    - name: commit the changes
      nclu:
          commit: yes

- hosts: core2
  become: yes
  remote_user: cumulus
  gather_facts: no
  vars:
    ansible_user: "cumulus"
    ansible_ssh_pass: "CumulusLinux!"
    ansible_become_pass: "CumulusLinux!"
```

```
tasks:
  - name: adiciona ip na VLAN 1 CORE2
    nclu:
      commands:
        - add vlan 10 ip address 10.0.0.2/24
        - add vlan 20 ip address 20.0.0.1/24
        - add vlan 30 ip address 30.0.0.1/24
        - add vlan 40 ip address 40.0.0.1/24

  - name: commit the changes
    nclu:
      commit: yes
```

Figura 73 - Criação de interfaces IP

```
everton_verissimo — root@NetworkAutomation-1: /etc/ansible/playbooks — telnet 172....
ml t@NetworkAutomation-1:/etc/ansible/playbooks# ansible-playbook ip_interface.ym

PLAY [core1] ***************************************************************

TASK [adiciona ip na VLAN 1 SW1] *******************************************
changed: [core1]

TASK [commit the changes] ***************************************************
ok: [core1]

PLAY [core2] ***************************************************************

TASK [adiciona ip na VLAN 1 SW2] *******************************************
changed: [core2]

TASK [commit the changes] ***************************************************
ok: [core2]

PLAY RECAP *****************************************************************
core1                      : ok=2    changed=1    unreachable=0    failed=0
core2                      : ok=2    changed=1    unreachable=0    failed=0

root@NetworkAutomation-1:/etc/ansible/playbooks#
root@NetworkAutomation-1:/etc/ansible/playbooks#
```

Fonte: Autor

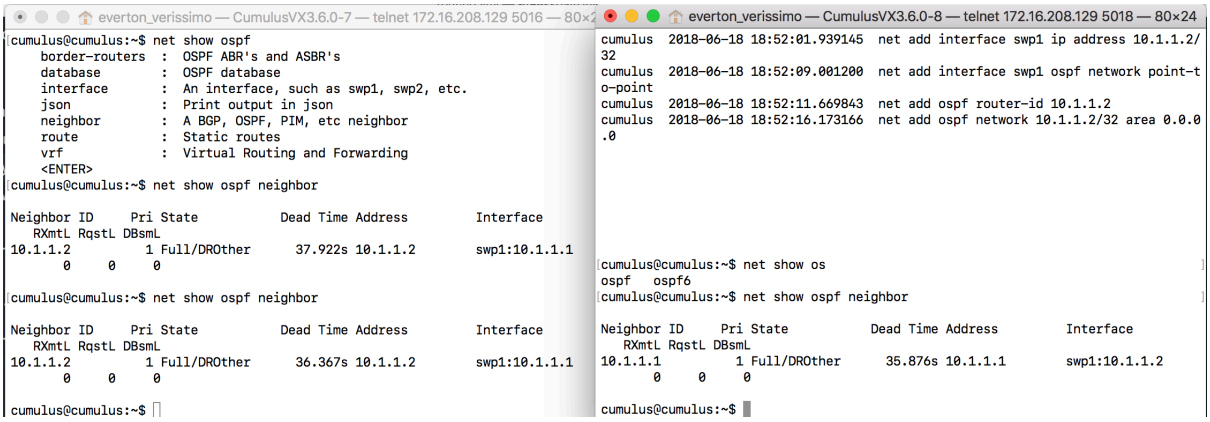
4.2.2. ROUTING

4˚ Passo – Para configuração do roteamento OSPF entre o Core1 e o Core2 foi criado um playbook chamado routing.yml, conforme a Figura 74.

```
---
- hosts: core1
  become: yes
  remote_user: cumulus
  gather_facts: no
  vars:
    ansible_user: "cumulus"
    ansible_ssh_pass: "CumulusLinux!"
    ansible_become_pass: "CumulusLinux!"

  tasks:
    - name: Configura OSPF entre CORE1 e CORE2
      nclu:
        commands:
          - add loopback lo ip address 10.1.1.1/32
          - add interface swp1 ip address 10.1.1.1/32
          - add interface swp1 ospf network point-to-point
          - add ospf router-id 10.1.1.1
          - add ospf network 10.1.1.1/32 area 0.0.0.0

    - name: commit the changes
      nclu:
          commit: yes

- hosts: core2
  become: yes
  remote_user: cumulus
  gather_facts: no
  vars:
```

```
  ansible_user: "cumulus"
  ansible_ssh_pass: "CumulusLinux!"
  ansible_become_pass: "CumulusLinux!"

tasks:
  - name: Configura OSPF entre CORE1 e CORE2
    nclu:
      commands:
        - add loopback lo ip address 10.1.1.2/32
        - add interface swp1 ip address 10.1.1.2/32
        - add interface swp1 ospf network point-to-point
        - add ospf router-id 10.1.1.2
        - add ospf network 10.1.1.2/32 area 0.0.0.0

  - name: commit the changes
    nclu:
      commit: yes
```

Figura 74 - Configuração OSPF



Fonte: Autor

4.2.3. TROUBLESHOOTING

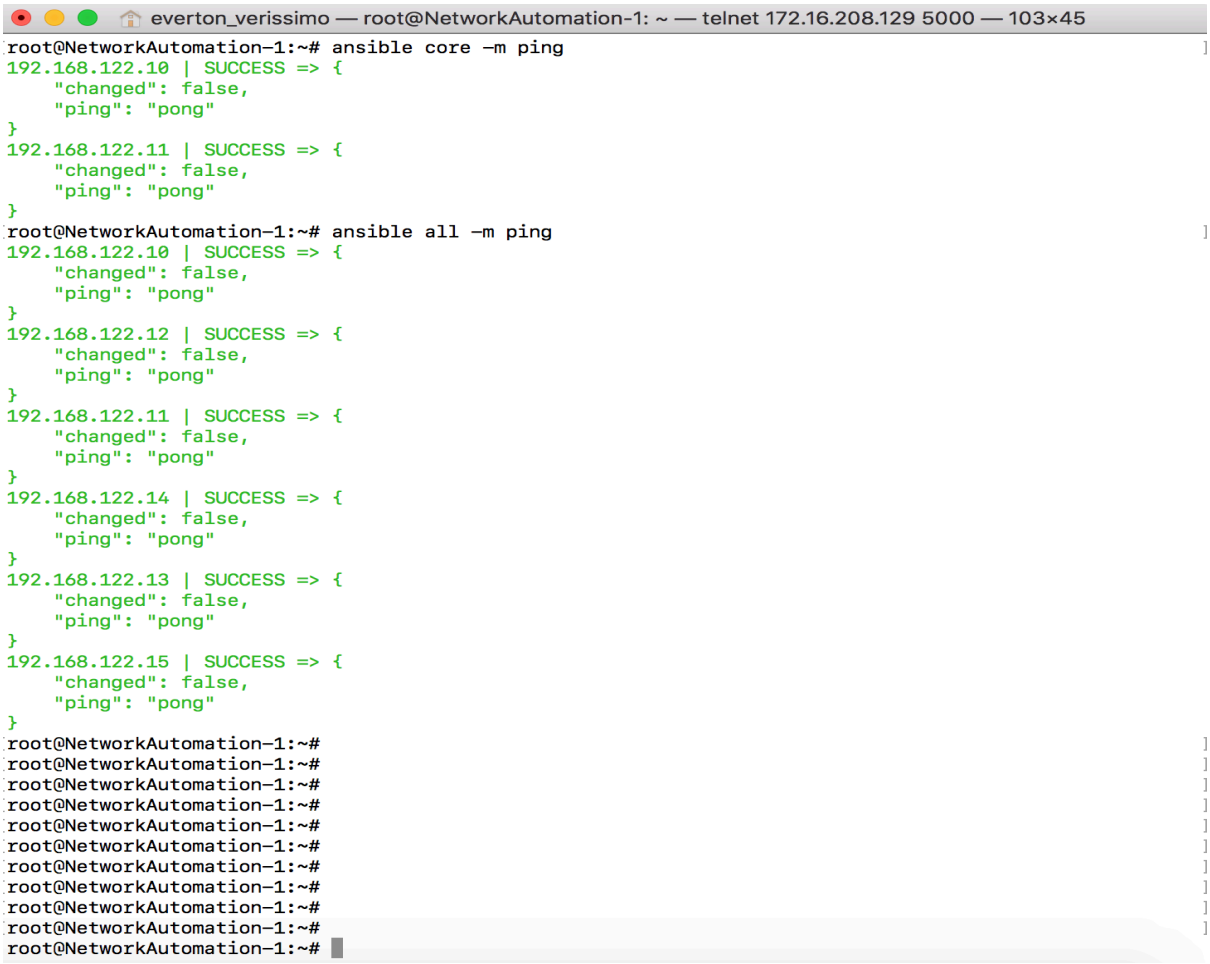
O *Troubleshooting* ou resolução de problemas é umas das tarefas que mais consome tempo em um ambiente de rede (BROMLEY, 2018). Abaixo serão demostradas algumas

ferramentas que podem pode contribuir para um automação de processo de *troubleshooting* reduzindo o tempo gasto.

4.2.3.1. PING

1˚ Passo – Uma vez instalada a ferramenta de automação pode executa o comando **ansible core -m ping** para teste com um grupo específico de switches ou **ansible all -m ping** para teste de conectividade com todos os switches, conforme a Figura 75.

Figura 75 - Teste Ping

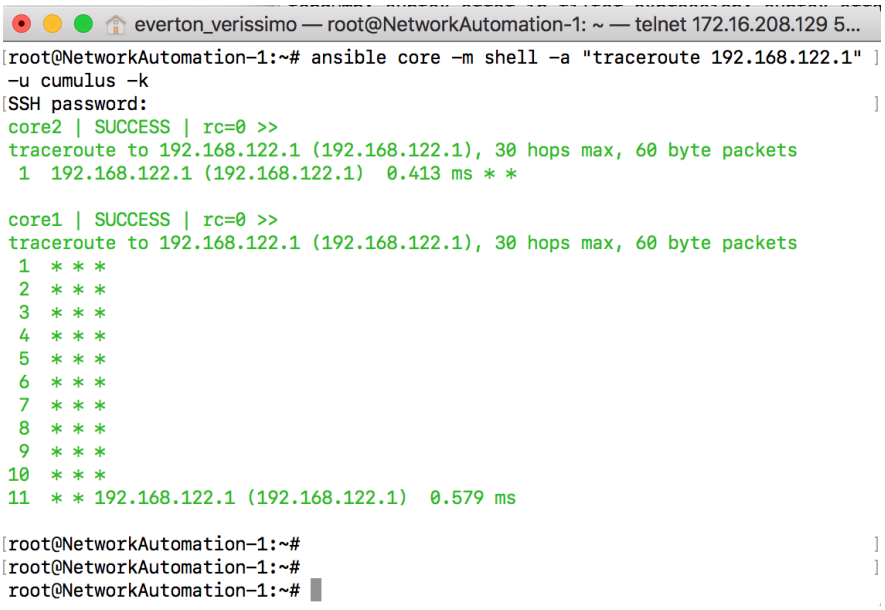


Fonte: Autor

4.2.3.2. TRACEROUTE

1˚ Passo – Outro comando que pode ser executado em massa, ou seja, destinado a um grupo de equipamentos é o comando traceroute. Neste exemplo utiliza-se o comando **ansible core -m shell -a "traceroute 192.168.122.1" -u cumulus -k , con**forme a Figura 76.

Figura 76 - Automatização comando traceroute

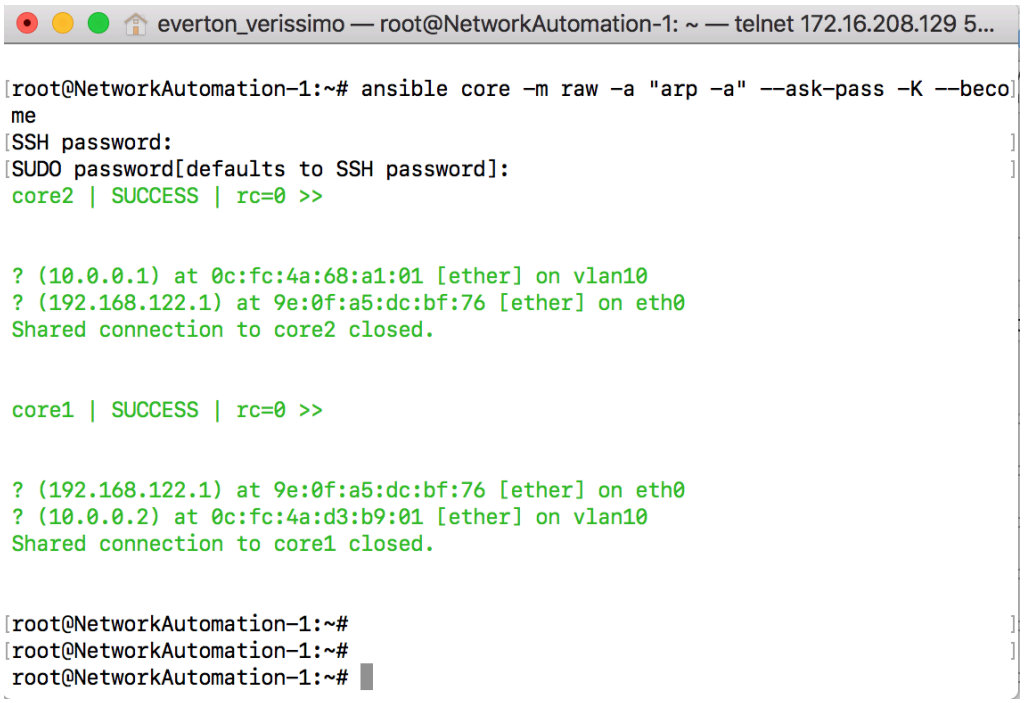


Fonte: Autor

4.2.3.3. ARP

1° Passo – Também pode-se utilizar o comando **ansible core -m raw -a "arp -a" --ask-pass -K --become** para automatizar a checagem de ARPs aprendidos pelo switch, conforme a Figura 77.

Figura 77 - Teste ARP

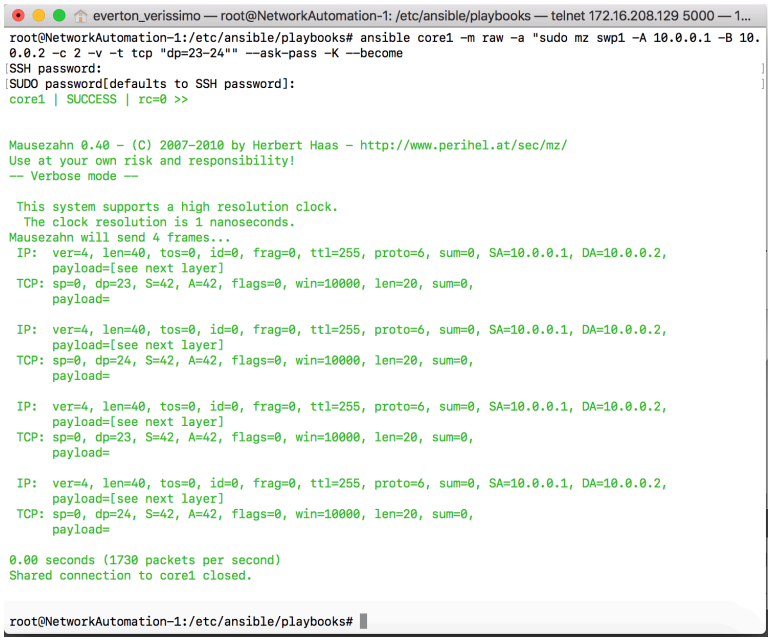


Fonte: Autor

4.2.3.4. MZ

1˚ Passo – Outra ferramenta que pode ser utilizada é o MZ, com ele pode-se gerar um tráfego na rede. Para o teste utiliza-se o comando **ansible core1 -m raw -a "sudo mz swp1 -A 10.0.0.1 -B 10.0.0.2 -c 2 -v -t tcp "dp=23-24"" --ask-pass -K –become, con**forme a Figura 78.

Figura 78 - Teste de geração de pacotes

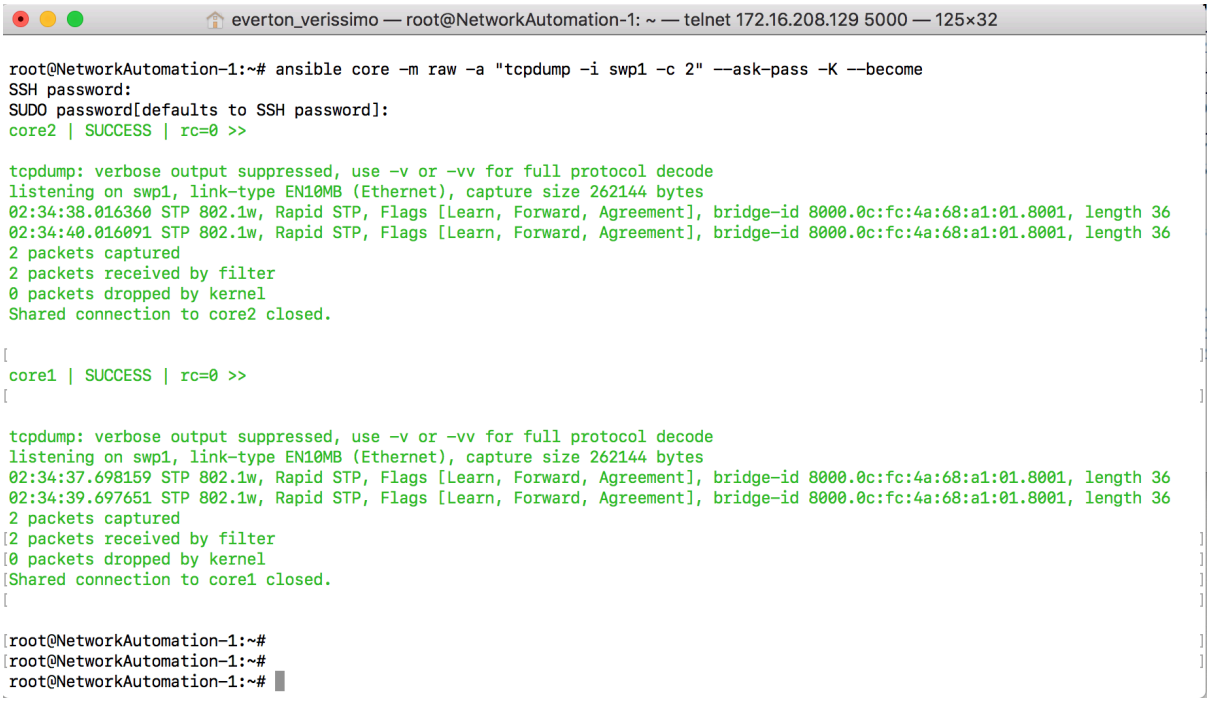


Fonte: Autor

4.2.3.5. TCPDUMP

1˚ Passo – Também pode-se fazer a análise de pacote utilizando o comando tcpdump. Neste passo utiliza-se o comando **ansible core -m raw -a "tcpdump -i swp1 -c 2" --ask-pass -K –become** podendo fazer a capturas de pacotes, conforme a Figura 79.

Figura 79 - Teste tcpdump



Fonte: Autor

4.2.4. BACKUPS E RESTAURAÇÃO

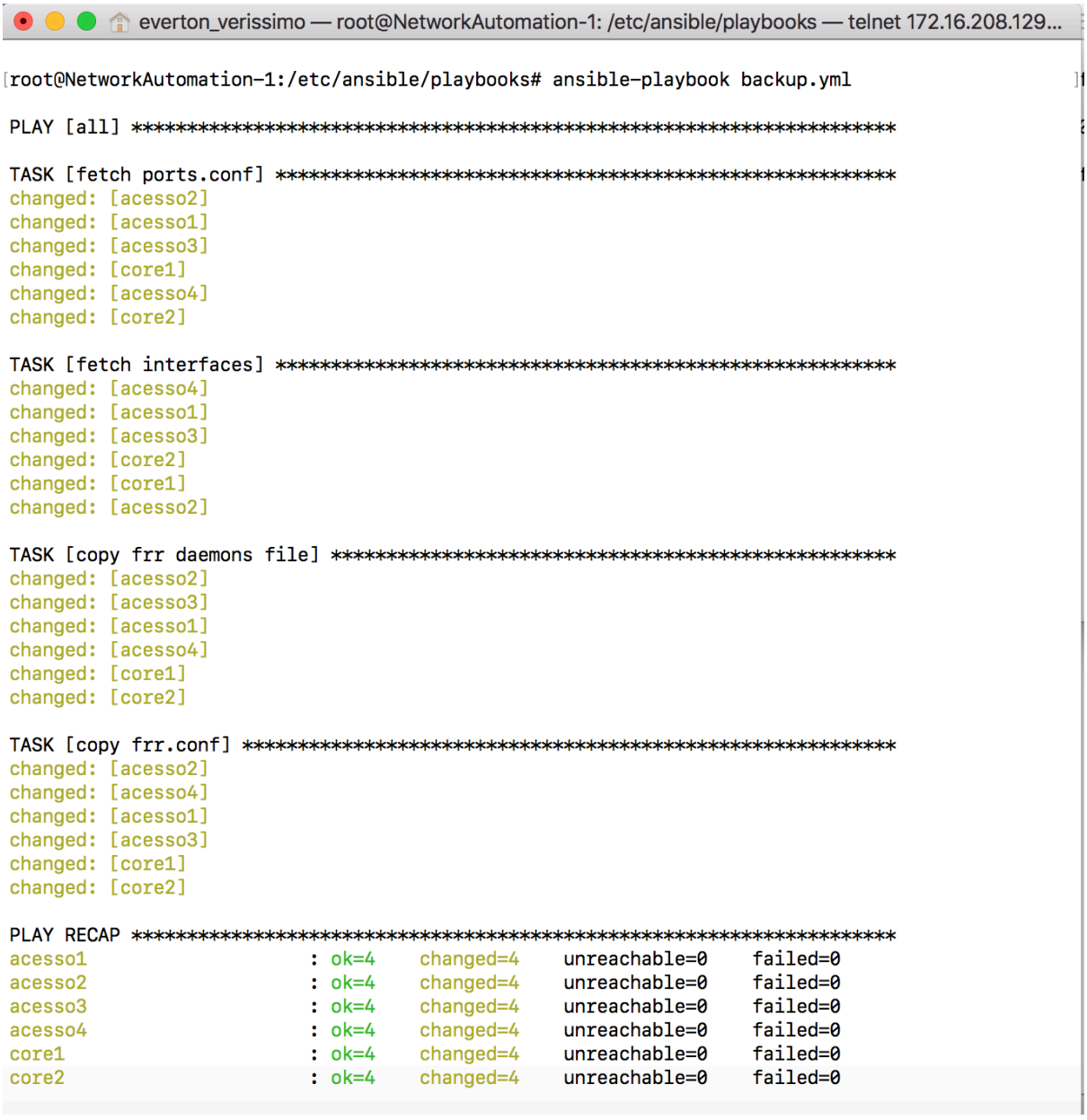
1° Passo – Para a a criação de backup foi criado um playbook escrito em YAML, denominado backup.yml. O arquivo salva os arquivos de configuração: /etc/cumulus/ports.conf, /etc/network/interfaces, /etc/frr/daemons e /etc/frr/frr.conf. Foi criado uma pasta com o nome playbooks no caminho /etc/ansible/. O comando para execução do playbook é **ansible-playbook backup.yml** Conforme a Figura 80.

Arquivo backup.yml

```
---
- hosts: all
  become: yes
  remote_user: cumulus
  gather_facts: no
  vars:
    ansible_user: "cumulus"
    ansible_ssh_pass: "CumulusLinux!"
    ansible_become_pass: "CumulusLinux!"

  tasks:
    - name: fetch ports.conf
      fetch: dest=backup/{{ansible_host}}/ports.conf src=/etc/cumulus/ports.conf flat=yes
```

```
- name: fetch interfaces
  fetch: dest=backup/{{ansible_host}}/interfaces src=/etc/network/interfaces flat=yes

- name: copy frr daemons file
  fetch: dest=backup/{{ansible_host}}/daemons src=/etc/frr/daemons flat=yes

- name: copy frr.conf
  fetch: dest=backup/{{ansible_host}}/frr.conf src=/etc/frr/frr.conf flat=yes
```

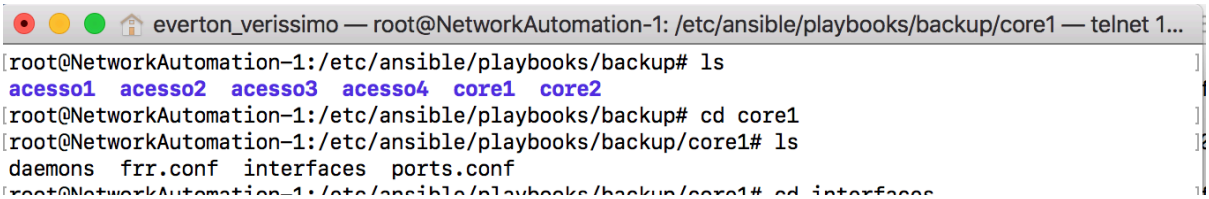
Figura 80 - Execução do playbook backup.yml



Fonte: Autor

2˚ Passo – Após executar o playbooks os arquivos de configuração serão salvos na pasta /etc/ansible/playbooks/backups, conforme a Figura 81.

Figura 81 - Criação de backup de todos os switches



Fonte: Autor

3˚ Passo – Para restaurar os arquivos de backup foi criado outro playbook chamado restore.yml, conforme a Figura 82.

```
---
- hosts: all
  become: yes
  remote_user: cumulus
  gather_facts: no
  vars:
    ansible_user: "cumulus"
    ansible_ssh_pass: "CumulusLinux!"
    ansible_become_pass: "CumulusLinux!"
  tasks:

    - name: copy ports.conf
      copy: src=backup/{{ansible_host}}/ports.conf dest=/etc/cumulus/
    - name: copy interfaces
      copy: src=backup/{{ansible_host}}/interfaces dest=/etc/network/
    - name: copy frr daemons
      copy: src=backup/{{ansible_host}}/daemons dest=/etc/frr/
    - name: copy frr conf
      copy: src=backup/{{ansible_host}}/frr.conf dest=/etc/frr/

    - name: reload switchd
      service: name=switchd state=restarted
    - name: reload networking
      command: ifreload -a
    - name: reload frr
      service: name=frr state=restarted
```

Figura 82 - Execução playbook restore.yml

```
everton_verissimo — root@NetworkAutomation-1: /etc/ansible/playbooks — telnet 172.16.208.129...
root@NetworkAutomation-1:/etc/ansible/playbooks# ansible-playbook restore.yml

PLAY [all] *********************************************************************

TASK [copy ports.conf] *********************************************************
ok: [acesso2]
ok: [acesso3]
ok: [acesso1]
ok: [acesso4]
ok: [core1]
ok: [core2]

TASK [copy interfaces] *********************************************************
ok: [acesso3]
ok: [core1]
ok: [acesso4]
ok: [acesso1]
ok: [acesso2]
ok: [core2]

TASK [copy frr daemons] ********************************************************
ok: [acesso2]
ok: [acesso1]
ok: [core1]
ok: [acesso4]
ok: [acesso3]
ok: [core2]

TASK [copy frr conf] ***********************************************************
ok: [acesso1]
ok: [acesso4]
ok: [acesso2]
ok: [core1]
ok: [acesso3]
ok: [core2]

TASK [reload switchd] **********************************************************
changed: [acesso1]
changed: [acesso3]
changed: [acesso2]
changed: [acesso4]
changed: [core1]
changed: [core2]
```

Fonte: Autor

4.2.5. AVALIAÇÃO DE DESEMPENHO

Para avaliação da redução no tempo de configuração todos os procedimentos foram realizados utilizando a automação com o Ansible e sem o uso de automação, também foram avaliados o tempo de provisionamento dos equipamentos tanto para atualização do sistema operacional de rede como para o provisionamento de configurações. Abaixo temos a Tabela 1 que demostrar os resultados dos testes.

Tabela 1 – Avaliação de desempenho da ferramenta de automação

	Configuração Manual	Automação com Ansible	Redução no tempo de configuração ou provisionamento
ONIE	900 segundos	120 segundos	87%
ZTP	600 segundos	120 segundos	80%
Switching	720 segundos	15 segundos	98%
Routing	900 segundos	20 segundos	98%
Ping	240 segundos	15 segundos	94%
Traceroute	300 segundos	15 segundos	95%
ARP	300 segundos	15 segundos	95%
MZ	300 segundos	15 segundos	95%
Backup	600 segundos	10 segundos	98%
Restore	900 segundos	10 segundos	99%
Total	5760 segundos	355 segundos	94%

Fonte: Autor

A Tabela 1 demostra a significativa redução no tempo de configuração de equipamentos, isso em um ambiente onde foram testados apenas 8 equipamentos de rede. Isso permite concluir que o uso de automação traz uma grande redução no tempo necessário para gerenciamento e configuração, reduzindo ainda possibilidade de erro e melhorando o tempo de recuperação em uma eventual falha na rede.

Isso atrelado a parâmetros que não foram avaliados neste teste com tempo de deslocamento e custos envolvidos de pessoal ou o tempo de indisponibilidade da rede, mostra o grande potencial da utilização de ferramentas de automação. Ainda podemos incluir outros benefícios com a gerência de configuração com versionamento e segurança nos acessos a estes dispositivos.

CAPÍTULO 5 – CONCLUSÕES FINAIS

5. CONCLUSÃO

Como foi exposto no decorrer deste trabalho, é verificado que a utilização de ferramentas de automação permite reduzir em 94% o tempo de implementação e configuração de uma rede campus. Isso juntos aos benefícios trazidos pela desagregação permite que a rede possa ser expandida sem a vinculação com determinados fabricantes. Podendo ainda em um futuro ser utilizando a arquitetura SDN sem que seja necessário a troca de hardware.

Os softwares de automação cumprem o seu papel e mostram-se bastante evoluídos em termos de funcionalidade. O Ansible permite um gerenciamento simplificado, facilitando as implementações de pequeno e médio porte.

Ainda existem muitos desafios para que as redes de computadores deixem de ser os gargalos em implementações de centro de dados ou redes campus, porém este trabalho teve o objeto de propor novas abordagem para o gerenciamento de rede campus que diferentemente dos centros de dados continua uma a sua estrutura monolítica é proprietário.

Este trabalho atingiu seus objetivos de conseguindo automatizar os processos baseado nas melhorias advindas de grandes centros de dados, melhorando a sua eficiência, flexibilidade e escalibilidade. Sem dúvida os novos profissionais de rede de computadores devem conhecer de automação, programação e assim como outros temas que antigamente eram dissociados do seguimento de rede.

O open networking não é apenas uma tendência, com apoio de grandes fabricantes e grandes provedores de serviço como o Facebook, mostra a importância de comunidades colaborativas como o OPC. Entretanto ainda existe espaço para melhorias, um exemplo seria a utilização de interfaces de rede controladora pelos ASIC para o provisionamento. A gerência feita através das interfaces de gerenciamento oneram o custo de uma solução de automação e provisionamento em uma rede campus, isso devido a necessidade de se ter uma rede paralela a rede de usuários dedicada exclusivamente para este fim.

5.1. SUGESTÕES PARA TRABALHOS FUTUROS

Para trabalhos futuros, sugere-se demostrar outras formas de implementação onde os equipamentos possam ser testados fisicamente. Podendo aumentar a complexidade dos testes e avaliar a performance dos equipamentos.

Open networking continua a evoluir, os fabricantes de ASIC já perceberam que a tendência e passaram a contribuir com projeto. O próprio Open Compute Project, possui algumas iniciativas como o Switch Abstraction Interface (SAI), que é liderado pela Microsoft (SUBRAMANIAM, 2016). O SAI é uma interface de abstração do ASIC, totalmente aberta que está sendo desenvolvida pelo OCP em conjunto com as fornecedoras de ASIC com a Broadcom e Mellanox. O objetivo é padronizar a forma com que os ASIC interagem com as camadas superiores. Com isso, temos a possibilidade de entender como realmente funciona o switch em seu mais baixo nível. Podendo aumentar as possibilidades de melhorar a performance, reduzir o consumo de energia, ou ainda fazer uma comutação e roteamento mais eficientes. O SAI traz novas perspectivas para open networking (OCP, 2017).

Outra sugestão para trabalhos futuros é a automação e provisionamento para os serviços em nuvem com a integração de diferentes áreas como infraestrutura, desenvolvimento e segurança.

REFERÊNCIAS

AL-SHAWI, M. CCDE Study Guide, Indianapolis: Cisco Press, 2016.

BENOKRAITIS, A. NetDevOps: Networking Methods with a DevOps Mindset, 2018, disponível em <https://cumulusnetworks.com/blog/netdevops-networking-methods-with-a-devops-mindset/>. Acesso em 1 de jun. 2018.

BOMBAL, D.; DUPONCHELLE, J. GNS3 Setup wizard with the GNS3 VM, disponível em <http://docs.gns3.com/1wdfvS-OlFfOf7HWZoSXMbG58C4pMSy7vKJFiKKVResc/index.html#h.seowsyt9e4j7>. Acesso em: 28 de mai. 2018.

BRATACH, P. Configuring switchd, 2016, disponível em < https://docs.cumulusnetworks.com/display/CL35/Configuring+switchd> Acesso em 14 de jun. 2018.

BROMLEY, K. 5 passos para otimizar a solução de problemas de rede, 2018, disponível em < http://computerworld.com.br/5-passos-para-otimizar-solucao-de-problemas-de-rede> Acesso em 15 de jun. 2018.

CHO, K. C. S. J. Efficient monitoring algorithm for fast news alert. IEEE Trans. Knowledge and Data Engineering, 2007. disponível em <https://oak.cs.ucla.edu/~cho/papers/tkde-0038-0106.pdf>. Acesso em 17 de jun. 2018.

CUMULUS NETWORKS. Cumulus Linux architecture, disponível em <https://cumulusnetworks.com/learn/web-scale-networking-resources/product-collateral/cumulus-linux-architecture/>. Acesso em: 28 de mai. 2018.

CUMULUS NETWORKS. Cumulus Linux 3.6.0 User Guide, 2018, disponível em <https://docs.cumulusnetworks.com/spaces/viewspace.action?key=DOCS&preview=/8357803/8358299/Cumulus%20Linux%203.6.0%20User%20Guide.pdf>. Acesso em: 14 de jun. 2018.

DUPONCHELLE, J.; GROSSMAN, J., GNS3 Technologies, disponível em <https://github.com/GNS3>. Acesso em: 28 de mai. 2018.

DOYLE, J. Clearing the fog around open switching terminology, 2015 disponível em < https://www.networkworld.com/article/2919599/cisco-subnet/clearing-the-fog-around-open-switching-terminology.html >. Acesso em: 28 de mai. 2018.

FRROUTING, Frr routing overview, disponível em <https://frrouting.org/user-guide/Overview.html#Overview >. Acesso em: 28 de mai. 2018.

GIESEN,Glaucio França; OLIVEIRA, Francine Cássia de, Abordagem SDN: uma mudança de paradigma na arquitetura de rede tradicional. In: VII SRST – SEMINÁRIO DE REDES E SISTEMAS DE TELECOMUNICAÇÕES, 2017, Minas Gerais. disponível em <https://www.inatel.br/biblioteca/pos-seminarios/seminario-de-redes-e-sistemas-de-

telecomunicacoes/2017-2/9641-abordagem-sdn-uma-mudanca-de-paradigma/file/>. Acesso em: 28 de mai. 2018.

GREENE, D. What is DevOps? Tech Crunch, 2015. Disponível em: <http://techcrunch.com/2015/05/15/what-is-devops/>. Acesso em: 28 de mai. 2018.

IP INFUSION, Carrier-grade NOS for bare metal switches OcNOS provides seamless transition from traditional networks, 2017, disponível em <https://www.ipinfusion.com/products/ocnos/> Acesso em 7 de jun. 2018.

LAMA, V. Software-Defined Networking and Open Networking: Understanding Foundational Concepts and Constructs v1.1, 2016, disponível em < http://en.community.dell.com/cfs-file/__key/telligent-evolution-components-attachments/13-4491-00-00-20-44-27-52/SDN-Foundational-Concepts-_2800_FINAL-v1.1_2900_.pdf?forcedownload=true > Acesso em 14 de jun. 2018.

MARSHALL, David, BEAVER, Stephen S., MCCARTY, Jason W. VMware ESX Essentials in the Virtual data Center. Auerbach Publications, 2009.

MENZEN, C. D. Estudo e Implementação de soluções para automação de dispositivos de rede, trabalho de conclusão de curso. (Tecnologia em Redes de Computadores), Universidade Federal de Santa Maria, 2015

METZLER, J. What's the Best Alternative to the Spanning Tree Protocol?, Leadership Discussion, 2011, disponível em < https://www.cisco.com/c/dam/en/us/solutions/collateral/data-center-virtualization/unified-fabric/stp_summary.pdf> Acesso em 13 de jun. 2018.

MILLER, L.C., Software-Defined Networking, Pluribus Networks Edition, New Jersey: John Wiley & Sons, Inc, 2016.

OCP. About OCP. Disponível em:<http://opencompute.org/about/>. Acesso em: 28 de mai. 2018.

OCP. OCP Networking. Disponível em:< http://opencompute.org/projects/networking/>. Acesso em: 28 de mai. 2018.

OCP. Networking Specs and Designs. 2016 Disponível em:< http://www.opencompute.org/wiki/Networking/SpecsAndDesigns#Hardware_Specs>. Acesso em: 14 de jun. 2018.

ONF, Open Networking Foundation, Leveraging Disaggregation to Build Innovative Open Source Solutions for Operator Network, 2018, disponível em < https://www.opennetworking.org>. Acesso em 28 de mai. 2018.

ONF, Open Networking Foundation, SDN architecture, 2014, disponível em < https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf> . Acesso em 14 de jun. 2018.

ONIE, Open Network Install Environment, 2018, disponível em <http://onie.org>. Acesso em: 28 de mai. 2018.

ONIE, Open Network Install Environment, 2018, disponível em < https://opencomputeproject.github.io/onie/overview/index.html#design-overview>. Acesso em: 31 de mai. 2018.

ONL, PROPOSAL: OCP COMMON LINUX SWITCH DISTRIBUTION, Apresentação, 2013, disponível em < https://opennetlinux.org/LinuxDistroOCP-November.pdf > Acesso em 7 de jun. 2018.

OTANI, N.; FIALHO Francisco A. P. TCC métodos e técnicas 2. ed. Florianópolis: Visual Books, 2011

PATTON, D. Cumulus Linux Network Command Line Utility, disponível em <https://cumulusnetworks.com/blog/cumulus-linux-network-command-line-utlility/>. Acesso em: 28 de mai. 2018.

PERZ, M. How Ansible Works, 2017, Vídeo disponível em < https://www.redhat.com/pt-br/about/videos/how-ansible-works>. Acesso em 1 de jun. 2018.

PICA8, White Box Networking in Remote and Branch Offices, Whitepaper, Palo Alto, 2017 Disponível em: < https://www.pica8.com/wp-content/uploads/white-box-networking-remote-branch-offices.pdf>. Acesso em: 14 de jun. 2018

PLURIBUS, Pluribus Open Netvisor Linux: Marrying SDN Automation with Server Style Management, 2015, disponível em <https://www.pluribusnetworks.com/blog/pluribus-open-netvisor-linux-marrying-sdn-automation-with-server-style-management/> Acesso em 7 de jun. 2018.

RAZA, A. puppet-vs.-chef-vs.-ansible-vs.-saltstack, 2016, disponível em <https://www.intigua.com/blog/puppet-vs.-chef-vs.-ansible-vs.-saltstack>. Acesso em 1 de jun. 2018.

REDHAT, whats-it-automation, 2018, disponível em <https://www.redhat.com/pt-br/topics/automation/whats-it-automation>. Acesso em 1 de jun. 2018.

REDHAT, whats-it-devops, 2018, disponível em < https://www.redhat.com/pt-br/topics/devops>. Acesso em 1 de jun. 2018

RODRÍGUEZ, F.L. Arquitetura e protótipo de uma rede SDN-OPENFLOW para provedor de serviço, Dissertação (Mestrado em Engenharia Elétrica), Universidade de Brasília, Brasília, 2014.

SHARMA, S. COYNE, B., DEVOPS, IBM Limited Edition, New Jersey: John Wiley & Sons, Inc, 2017.

SHEN, Y. Global Internet Routing Table Reaches 512k Milestone, 2014, disponível em < https://blogs.cisco.com/sp/global-internet-routing-table-reaches-512k-milestone>. Acesso em: 18 de jun. 2018.

SOUSA, M. SILVA, J. SOUZA, W. CABRAL, Y. ROCHA, U. OLIMPIO, T. Simuladores e Emuladores de Rede para o Projeto e Solução de Problemas em Ambientes de Produção. revista de tecnologia da informação e comunicação, 2016, Paraíba.

SUBRAMANIAM, K. Microsoft showcases “Software for Open Networking in the Cloud (SONiC), 2016, disponível em < https://azure.microsoft.com/pt-br/blog/microsoft-showcases-software-for-open-networking-in-the-cloud-sonic/>. Acesso em: 28 de mai. 2018.

TANENBAUM, Andrew .S, Redes de computadores, 4ª edição, Rio de Janeiro: Elsevie, 2003.

TOGHRAEE, R. Datacenter 10G , 40G Leaf Spine Switch Buying Guide, 2016, disponível em <http://www.arpaware.com/blog/arpaware-technology-blog-1/post/datacenter-10g-40g-leaf-spine-switch-buying-guide-2016-june-7> Acesso em 14 de jun. 2018.

U-BOOT, Universal Boot Loader, Manual, disponível em < https://www.denx.de/wiki/view/DULG/Manual > . Acesso em 31 de mai. 2018.

WILSON, N. The rise of the DevOps culture, and why it's important. Tech Radar, 19 jan. 2015. Disponível em: <http://www.techradar.com/news/world-of- tech/management/the-rise-of-the-devops-culture-and-why-it-s-important-1281130>. Acesso em: 28 de mai. 2018.

WELLS, T. Network Command Line Utility – NCLU, 2018, disponível em < https://docs.cumulusnetworks.com/display/DOCS/Network+Command+Line+Utility+-+NCLU> Acesso em 14 de jun. 2018.

APÊNDICES

Arquivo /etc/default/isc-dhcp-server

```
# Defaults for isc-dhcp-server initscript
# sourced by /etc/init.d/isc-dhcp-server
# installed at /etc/default/isc-dhcp-server by the maintainer scripts

#
# This is a POSIX shell fragment
#

# Path to dhcpd's config file (default: /etc/dhcp/dhcpd.conf).
#DHCPD_CONF=/etc/dhcp/dhcpd.conf

# Path to dhcpd's PID file (default: /var/run/dhcpd.pid).
#DHCPD_PID=/var/run/dhcpd.pid

# Additional options to start dhcpd with.
#       Don't use options -cf or -pf here; use DHCPD_CONF/ DHCPD_PID instead
#OPTIONS=""

# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?

# Path to dhcpd's PID file (default: /var/run/dhcpd.pid).
#DHCPD_PID=/var/run/dhcpd.pid

# Additional options to start dhcpd with.
#       Don't use options -cf or -pf here; use DHCPD_CONF/ DHCPD_PID instead
#OPTIONS=""

# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
#       Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACES="eth0"
```

==========

Arquivo /etc/dhcp/dhcpd.conf

```
#
# Sample configuration file for ISC dhcpd for Debian
#
# Attention: If /etc/ltsp/dhcpd.conf exists, that will be used as
# configuration file instead of this file.
#
#

# The ddns-updates-style parameter controls whether or not the server will
# attempt to do a DNS update when a lease is confirmed. We default to the
# behavior of the version 2 packages ('none', since DHCP v2 didn't
# have support for DDNS.)
```

```
ddns-update-style none;

# option definitions common to all supported networks...
#option domain-name "example.org";
#option domain-name-servers ns1.example.org, ns2.example.org;

option domain-name "teste.lab";
option domain-name-servers NetworkAutomation-1.teste.lab;

default-lease-time 600;
max-lease-time 7200;

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
#authoritative;
authoritative;

# Use this to send dhcp log messages to a different log file (you also
# have to hack syslog.conf to complete the redirection).
log-facility local7;

# No service will be given on this subnet, but declaring it helps the
# DHCP server to understand the network topology.

#subnet 10.152.187.0 netmask 255.255.255.0 {
#}

# This is a very basic subnet declaration.

#subnet 10.254.239.0 netmask 255.255.255.224 {
#  range 10.254.239.10 10.254.239.20;
#  option routers rtr-239-0-1.example.org, rtr-239-0-2.example.org;
#}

# This declaration allows BOOTP clients to get dynamic addresses,
# which we don't really recommend.

#subnet 10.254.239.32 netmask 255.255.255.224 {
#  range dynamic-bootp 10.254.239.40 10.254.239.60;
#  option broadcast-address 10.254.239.31;
#  option routers rtr-239-32-1.example.org;
#}

option cumulus-provision-url code 239 = text;

subnet 192.168.122.0 netmask 255.255.255.0 {
  range 192.168.122.10 192.168.122.20;
  option subnet-mask 255.255.255.0;
  option routers 192.168.122.1;
  option broadcast-address 192.168.122.255;
```

```
  option cumulus-provision-url "http://192.168.122.1/demo.sh";
}

host core1 {
        hardware ethernet 0c:fc:4a:68:a1:00;
        fixed-address 192.168.122.10;
        option host-name "core1";
}

host core2 {
        hardware ethernet 0c:fc:4a:d3:b9:00;
        fixed-address 192.168.122.11;
        option host-name "core2";
}

host acesso1 {
        hardware ethernet 0c:fc:4a:4c:d1:00;
        fixed-address 192.168.122.12;
        option host-name "acesso1";
}

host acesso2 {
        hardware ethernet 0c:fc:4a:42:6f:00;
        fixed-address 192.168.122.13;
        option host-name "acesso2";
}

host acesso3 {
        hardware ethernet 0c:fc:4a:d7:c9:00;
        fixed-address 192.168.122.14;
        option host-name "acesso3";
}

host acesso4 {
        hardware ethernet 0c:fc:4a:5f:12:00;
        fixed-address 192.168.122.15;
        option host-name "acesso4";
}

# A slightly different configuration for an internal subnet.
#subnet 10.5.5.0 netmask 255.255.255.224 {
#  range 10.5.5.26 10.5.5.30;
#  option domain-name-servers ns1.internal.example.org;
#  option domain-name "internal.example.org";
#  option subnet-mask 255.255.255.224;
#  option routers 10.5.5.1;
#  option broadcast-address 10.5.5.31;
#  default-lease-time 600;
#  max-lease-time 7200;
#}
```

```
# Hosts which require special configuration options can be listed in
# host statements.   If no address is specified, the address will be
# allocated dynamically (if possible), but the host-specific information
# will still come from the host declaration.

#host passacaglia {
#  hardware ethernet 0:0:c0:5d:bd:95;
#  filename "vmunix.passacaglia";
#  server-name "toccata.fugue.com";
#}

# Fixed IP addresses can also be specified for hosts.   These addresses
# should not also be listed as being available for dynamic assignment.
# Hosts for which fixed IP addresses have been specified can boot using
# BOOTP or DHCP.   Hosts for which no fixed address is specified can only
# be booted with DHCP, unless there is an address range on the subnet
# to which a BOOTP client is connected which has the dynamic-bootp flag
# set.
#host fantasia {
#  hardware ethernet 08:00:07:26:c0:a5;
#  fixed-address fantasia.fugue.com;
#}

# You can declare a class of clients and then do address allocation
# based on that.   The example below shows a case where all clients
# in a certain class get addresses on the 10.17.224/24 subnet, and all
# other clients get addresses on the 10.0.29/24 subnet.

#class "foo" {
#  match if substring (option vendor-class-identifier, 0, 4) = "SUNW";
#}

#shared-network 224-29 {
#  subnet 10.17.224.0 netmask 255.255.255.0 {
#    option routers rtr-224.example.org;
#  }
#  subnet 10.0.29.0 netmask 255.255.255.0 {
#    option routers rtr-29.example.org;
#  }
#  pool {
#    allow members of "foo";
#    range 10.17.224.10 10.17.224.250;
#  }
#  pool {
#    deny members of "foo";
#    range 10.0.29.10 10.0.29.230;
#  }
#}
```

==========

nano /etc/apache/apache2.conf

```
Mutex file:${APACHE_LOCK_DIR} default


PidFile ${APACHE_PID_FILE}

Timeout 300

KeepAlive On

MaxKeepAliveRequests 100

KeepAliveTimeout 5

User ${APACHE_RUN_USER}
Group ${APACHE_RUN_GROUP}

HostnameLookups Off

ErrorLog ${APACHE_LOG_DIR}/error.log

LogLevel warn

IncludeOptional mods-enabled/*.load
IncludeOptional mods-enabled/*.conf

Include ports.conf

<Directory />
	Options FollowSymLinks
	AllowOverride None
	Require all denied
</Directory>

<Directory /usr/share>
	AllowOverride None
	Require all granted
</Directory>

<Directory /var/www/>
	Options Indexes FollowSymLinks
	AllowOverride None
	Require all granted
</Directory>

AccessFileName .htaccess
```

```
<FilesMatch "^\.ht">
Require all denied
</FilesMatch>

LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" $
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combin$
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

IncludeOptional conf-enabled/*.conf

IncludeOptional sites-enabled/*.conf

IncludeOptional conf-enabled/*.conf

IncludeOptional sites-enabled/*.conf

ServerName NetworkAutomation-1
```

==========

nano /etc/hosts

```
127.0.1.1       NetworkAutomation-1
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

192.168.122.1   NetworkAutomation-1
192.168.122.10 core1
192.168.122.11 core2
192.168.122.12 acesso1
192.168.122.13 acesso2
192.168.122.14 acesso3
192.168.122.15 acesso4
```

==========

nano /etc/ansible/ansible.cfg

```
[defaults]
inventory = /etc/ansible/hosts
```

==========

nano /etc/ansible/hosts

[core]
core1
core2

[acesso]
acesso1
acesso2
acesso3
acesso4
==========