MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

EDGAR LUIZ SIQUEIRA BARBOSA JÚNIOR
GABRIEL STEFANO PALERMO

# AMBIENTE COMPUTACIONAL PARA SIMULAÇÃO DE INFRAESTRUTURAS CRÍTICAS

Rio de Janeiro
2019

INSTITUTO MILITAR DE ENGENHARIA

EDGAR LUIZ SIQUEIRA BARBOSA JÚNIOR
GABRIEL STEFANO PALERMO

# AMBIENTE COMPUTACIONAL PARA SIMULAÇÃO DE INFRAESTRUTURAS CRÍTICAS

Projeto de Fim de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Engenheiro de Computação.

Orientador: TC Anderson Fernandes Pereira dos Santos - D.Sc.
Co-Orientadora: Prof$^{a}$. Raquel Coelho Gomes Pinto - D.Sc.

Rio de Janeiro
2019

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 - Praia Vermelha
Rio de Janeiro - RJ CEP 22290-270



004.69 Barbosa Júnior, Edgar Luiz Siqueira
S586e Ambiente Computacional para Simulação de Infraestruturas Críticas / Edgar Luiz Siqueira Barbosa Júnior , Gabriel Stefano Palermo, orientado por Anderson Fernandes Pereira dos Santos e Raquel Coelho Gomes Pinto - Rio de Janeiro: Instituto Militar de Engenharia, 2019.

68p.: il.

Projeto de Fim de Curso (graduação) - Instituto Militar de Engenharia, Rio de Janeiro, 2019.

1. Curso de Graduação em Engenharia de Computação - projeto de fim de curso. 1. Virtualização. 2. SCADA. 3. OpenStack. I. dos Santos, Anderson Fernandes Pereira . II. Pinto, Raquel Coelho Gomes . III. Título. IV. Instituto Militar de Engenharia.

INSTITUTO MILITAR DE ENGENHARIA

EDGAR LUIZ SIQUEIRA BARBOSA JÚNIOR
GABRIEL STEFANO PALERMO

# AMBIENTE COMPUTACIONAL PARA SIMULAÇÃO DE INFRAESTRUTURAS CRÍTICAS

Projeto de Fim de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Engenheiro de Computação.

Orientador: TC Anderson Fernandes Pereira dos Santos - D.Sc.

Co-Orientadora: Profª. Raquel Coelho Gomes Pinto - D.Sc.

Aprovado em 8 de Outubro de 2019 pela seguinte Banca Examinadora:

TC Anderson Fernandes Pereira dos Santos - D.Sc. do IME - Presidente

Profª. Raquel Coelho Gomes Pinto - D.Sc. do IME

TC Julio Cesar Duarte - D.Sc. do IME

Maj Leandro de Mattos Ferreira - M.Sc. do IME

Rio de Janeiro
2019

Ao Instituto Militar de Engenharia, por proporcionar um indescritível crescimento pessoal e profissional.

## AGRADECIMENTOS

Primeiramente gostaríamos de agradecer a todos os que contribuíram, diretamente ou indiretamente, para o desenvolvimento desse projeto. Em especial aos professores TC Anderson e Raquel, que nos orientaram no melhor caminho e sempre estiveram disponíveis para nos auxiliar e garantir o sucesso do trabalho. Também aos professores TC Duarte e Maj Leandro Ferreira que contribuiram bastante com o desenvolvimento do projeto com seus conselhos durante as avaliações parciais.

Agradecemos a todos que nos apoiaram ao longo dos últimos cinco anos e acompanharam nossas trajetórias no Instituto Militar de Engenharia.

“ *If you go back a few hundred years, what we take for granted today would seem like magic – being able to talk to people over long distances, to transmit images, flying, accessing vast amounts of data like an oracle. These are all things that would have been considered magic a few hundred years ago.* ”

ELON MUSK

## SUMÁRIO

LISTA DE ILUSTRAÇÕES ..... 9

LISTA DE TABELAS ..... 11

LISTA DE SIGLAS ..... 12

**1 INTRODUÇÃO** ..... 15
1.1 Motivação ..... 16
1.1.1 Vulnerabilidade e necessidade de segurança ..... 16
1.1.2 Proteção das infraestruturas críticas ..... 17
1.2 Objetivo ..... 18
1.3 Justificativa ..... 18
1.4 Metodologia ..... 19
1.5 Estrutura ..... 19

**2 CONCEITOS FUNDAMENTAIS** ..... 21
2.1 Infraestruturas críticas ..... 21
2.2 ICS ..... 21
2.2.1 PLC ..... 22
2.2.2 SCADA ..... 22
2.3 Virtualização ..... 23
2.3.1 Hipervisores ..... 24
2.3.2 Gêmeo digital ..... 25
2.4 Computação em nuvem ..... 25
2.4.1 IaaS ..... 27
2.4.2 PaaS ..... 27
2.4.3 SaaS ..... 27

**3 FERRAMENTAS E SOFTWARES** ..... 29
3.1 OpenStack ..... 29
3.1.1 Keystone ..... 30
3.1.2 Horizon ..... 31
3.1.3 Nova ..... 31
3.1.4 Neutron ..... 31

3.1.5 Glance ... 31
3.1.6 Swift ... 32
3.2 Rapid SCADA ... 32
3.2.1 Recursos ... 32

**4 PROPOSTA** ... 34
4.1 Ambiente ... 34
4.2 Arquitetura ... 34
4.3 Virtualização ... 36
4.3.1 Máquinas ... 36
4.3.2 Redes e firewall ... 36

**5 IMPLEMENTAÇÃO DO SIMULADOR DE INFRAESTRUTURA CRÍTICA** ... 38
5.1 Considerações iniciais ... 38
5.2 Camada de gerenciamento de operações ... 39
5.2.1 Servidor web ... 39
5.2.2 Servidor de banco de dados ... 40
5.2.3 Desktop ... 41
5.3 Camada de supervisão e monitoramento ... 41
5.3.1 Local HMI ... 41
5.3.2 Servidor SCADA ... 42
5.4 Camada de controle ... 43
5.4.1 PLC ... 43
5.5 Grupos de segurança ... 44
5.6 Arquitetura de redes ... 46

**6 TESTES E VALIDAÇÕES** ... 50
6.1 Servidores web e banco de dados ... 50
6.2 Tráfego de rede ... 51

**7 CONCLUSÃO** ... 54

**8 REFERÊNCIAS BIBLIOGRÁFICAS** ... 55

**9 APÊNDICES** ... 59
9.1 APÊNDICE 1: Instalação do OpenStack ... 60

9.2 APÊNDICE 2: Criação de máquinas virtuais ..... 65

## LISTA DE ILUSTRAÇÕES


FIG.2.1 Arquitetura SCADA com o padrão ISA-95 extraído de (ENISA, 2016). ..... 23
FIG.2.2 Hipervisores dos tipos 1 e 2. ..... 24
FIG.2.3 Tipos de computação em nuvem, adaptado de Adeeb (2014). ..... 26

FIG.3.1 Diagrama de recursos do OpenStack. ..... 29
FIG.3.2 Arquitetura dos projetos do OpenStack. ..... 30
FIG.3.3 Arquitetura do Rapid SCADA, extraído de Rapid SCADA (2019). ..... 33

FIG.4.1 Arquitetura proposta para implementação. ..... 35

FIG.5.1 Configurações do servidor *web*. ..... 40
FIG.5.2 Configurações do servidor de banco de dados. ..... 41
FIG.5.3 Tabela de registro de eventos de acesso. ..... 41
FIG.5.4 Configuração dos canais de comunicação utilizando o Rapid SCADA. 42
FIG.5.5 Configuração de dispositivos utilizando o Rapid SCADA. ..... 43
FIG.5.6 Interface *web* do Rapid SCADA. ..... 43
FIG.5.7 Emulador OpenPLC. ..... 44
FIG.5.8 *Firewall* implementado com grupos de segurança. ..... 45
FIG.5.9 Regras do grupo de segurança *supervisory-lan-fw*. ..... 46
FIG.5.10 Endereço da *bridge* 'br-ex'. ..... 47
FIG.5.11 Visualização em grafo da rede no Horizon. ..... 48
FIG.5.12 Visualização em topologia consolidada da rede no Horizon. ..... 49
FIG.5.13 Visualização em topologia completa da rede no Horizon. ..... 49

FIG.6.1 Página PHP para o servidor *web*. ..... 50
FIG.6.2 Servidor *web* conectado ao servidor de banco de dados. ..... 51
FIG.6.3 Tráfego de rede para o servidor *web*. ..... 52
FIG.6.4 Tráfego de rede para o servidor MySQL. ..... 53
FIG.6.5 Tráfego de rede com protocolo MODBUS/TCP. ..... 53

FIG.9.1 Arquivo de configuração mínima para instalação. ..... 61
FIG.9.2 Conclusão da instalação do OpenStack. ..... 62
FIG.9.3 Tela de login do *dashboard* do OpenStack. ..... 63

FIG.9.4 Informação do sistema do OpenStack. . . . . 64

FIG.9.5 Criação de imagem no OpenStack. . . . . 65
FIG.9.6 Criação da instância no Horizon. . . . . 67
FIG.9.7 Seleção de imagem para criação de instância no Horizon. . . . . 67
FIG.9.8 Máquina virtual criada na página de instâncias no Horizon. . . . . 67
FIG.9.9 Configuração de rede em máquina virtual Windows. . . . . 68

## LISTA DE TABELAS

TAB.2.1 Recursos gerenciados pela nuvem, adaptado de Goran et al. (2014) .... 28

TAB.4.1 Máquinas virtuais na arquitetura proposta. ..... 37

TAB.5.1 Componentes da camada de gerenciamento de operações. ..... 39
TAB.5.2 Políticas de *firewall* dos grupos de segurança. ..... 46
TAB.5.3 Especificação dos endereços de rede. ..... 47

## LISTA DE SIGLAS

| | |
|---|---|
| API | Application Programming Interface |
| CERT | Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança |
| CPNI | Centre for the Protection of National Infrastructure |
| CPU | Central Process Unit |
| DHCP | Dynamic Host Configuration Protocol |
| DMZ | Demilitarized Zone |
| EPCIP | European Programme for Critical Infrastructure Protection |
| HMI | Human Machine Interface |
| HTTP | Hypertext Transfer Protocol |
| IaaS | Infrastructure as a Service |
| ICS | Industrial Control Systems |
| IP | Internet Protocol |
| LAN | Local Area Network |
| NASA | National Aeronautics and Space Administration |
| NIPP | National Infrastructure Protection Plan |
| PaaS | Platform as a Service |
| PLC | Programmable Logic Controller |
| SaaS | Software as a Service |
| SCADA | Supervisory Control and Data Acquisition |
| TCP | Transmission Control Protocol |
| TI | Tecnologia da Informação |
| UDP | User Datagram Protocol |

## RESUMO

Infraestruturas críticas são de grande importância para o bom funcionamento da sociedade. Pequenas falhas em suas operações são suficientes para causar grandes prejuízos financeiros e comprometer várias outras atividades. Estas podem ser ocasionadas tanto por agentes naturais quanto por ataques intencionais. Em um mundo cada vez mais conectado e integrado, se faz necessário mapear vulnerabilidades e proteger estes ativos, instalações e sistemas não apenas de ameaças físicas, mas também de ameaças cibernéticas.

Com os avanços tecnológicos, tornou-se possível desenvolver novas ferramentas para teste, simulação e virtualização. Ao longo deste projeto foi construído um ambiente computacional voltado para infraestruturas críticas. Este permite a simulação e testes de ativos de TIC em um espaço virtualizado. Com isso é possível preservar a integridade das infraestruturas físicas, além de realizar avaliações e testes sem interromper ou prejudicar o funcionamento dos sistemas reais.

## ABSTRACT

Critical infrastructures are of great importance in order to have a well-functioning society. Even minor failures in its operations are sufficient to cause large financial losses and jeopardize various other activities. These malfunctions can be caused by both natural and intentional human attacks. In an increasingly connected and integrated world, vulnerabilities need to be explored and these assets, facilities and systems must be protected not only from physical but also from cyber threats.

With technological advances, it has become possible to develop new tools for testing, simulation and virtualization. Throughout this project a computational environment for critical infrastructures was built. This enables the simulation and testing of CIT assets in a virtualized environment. This also allows to preserve the integrity of physical infrastructures, while performing evaluations and tests without disrupting the operation of the real systems.

# 1 INTRODUÇÃO

Infraestruturas críticas são ativos (instalações, serviços, bens ou sistemas) essenciais para o bom funcionamento da sociedade. São exemplos de infraestruturas críticas sistemas relacionados à geração e transmissão de energia, fornecimento de água, telecomunicações e informação, finanças e saúde. Serviços prestados pelas infraestruturas de transporte e finanças também são exemplos de infraestruturas críticas. A interrupção destas pode resultar em significativo impacto social, ambiental, econômico e político, além de provocar risco à segurança da sociedade (CHURCH et al., 2004).

Por serem indispensáveis, faz-se necessário buscar formas de minimizar suas vulnerabilidades e os riscos de descontinuidade da prestação desses serviços. Fatores que originam interrupções podem estar relacionados tanto a fenômenos e catástrofes naturais, devido à fragilidade das estruturas físicas, quanto à ação humana. Neste caso, vulnerabilidades e falhas de processos, sistemas e operações podem permitir ataques que comprometam o fornecimento dos serviços.

A operação destes sistemas não deve ser tratada de forma isolada. As infraestruturas críticas podem apresentar interdependências umas com as outras, de forma que falhas de um sistema afetam diretamente o funcionamento de outros. Segundo Rinaldi et al. (2001), dependências nessas infraestruturas podem ser classificadas como:

- físicas (um ativo depende da saída de outro para operar);
- cibernéticas (o acesso a dados necessários para o funcionamento é comprometido sem uma infraestrutura de informação operante); ou
- geográficas (o dano a uma instalação afeta diretamente outra devido à proximidade).

A simulação computacional tem como objetivo modelar o comportamento de um sistema ao longo do tempo e é constantemente utilizada para criação de ambientes virtuais a fim de imitar o funcionamento de um processo do mundo real. Com a simulação, é possível projetar um modelo computacional de um sistema real e conduzir experimentos com este com o propósito de entender seu comportamento e avaliar estratégias para sua operação. O desenvolvimento de um simulador de ambiente de infraestrutura crítica torna-se, portanto, essencial na tentativa de assegurar a qualidade e continuidade do

serviço oferecido, bem como para verificar se o sistema está preparado contra possíveis imprevistos e falhas em seu funcionamento.

## 1.1 MOTIVAÇÃO

### 1.1.1 VULNERABILIDADE E NECESSIDADE DE SEGURANÇA

Em diversas ocasiões, infraestruturas críticas foram alvos de falhas e ataques cibernéticos que comprometeram parcial ou integralmente seus funcionamentos. A interrupção dos serviços comprometem o funcionamento de inúmeras outras atividades, que são dependentes destas, além de provocar elevados prejuízos financeiros. A seguir, encontram-se listados alguns casos e seus impactos.

- Usina Hidrelética de Itaipu

  Em Novembro de 2009, devido a condições meteorológicas adversas, houve um curto circuito em três linhas de transmissão ocasionando o desligamento automático de 20 turbinas na Usina Hidrelética de Itaipu. Isso afetou 18 estados, deixando aproximadamente 60 milhões de pessoas sem energia elétrica por um período de 5 horas. Como medida emergencial, houve um custo de cerca de 470 milhões de reais para utilização de fontes alternativas de energia (SODRÉ, 2015).

- Centrífugas de urânio do Irã

  Descoberto em Junho de 2010 pela empresa VirusBlokAda, o *Stuxnet* é um *worm* de computador feito para atacar especificamente o SCADA desenvolvido pela empresa Siemens para o controle automático das centrífugas. Esse vírus atua no controle e monitoramento de processos industriais sendo capaz de reprogramar um Controle Lógico Programável (PLC - Programmable Logic Controller) e esconder as mudanças. O ataque ocorreu na Siemens no Irã entre 2005 a 2010 e fez com que as centrífugas variassem de velocidade, sendo aceleradas aos poucos, o que dificultou o reconhecimento da causa. O vírus danificou cerca de 10% das centrífugas de urânio do Irã (INVASÃO, 2019).

- Fornecimento de energia na Ucrânia

  No final de 2015, uma interrupção no fornecimento de energia na Ucrânia provocado por um *malware*, chamado de *BlackEnergy*, comprometeu cerca de 20% da rede e atingiu milhares de pessoas. Anteriormente, era usado apenas para espionagem, mas ao ser atualizado passou a ser efetivo em tomar controle remotamente e desligar

sistemas inteiros de grandes companhias. O ataque teria ocorrido devido a infecção de documentos do Microsoft Office (TECMUNDO, 2016).

- Guerra Cibernética

  Devido ao risco de exploração de eventuais vulnerabilidades por ameaças cibernéticas, há um aumento na preocupação dos órgãos de Estado e de governo envolvidos na segurança e na defesa nacional (DE CARVALHO, 2019). Devido a isso, o Brasil tem aumentado sua defesa cibernética a fim de proteger os sistemas de informação de interesse da Defesa Nacional, obter dados para a produção de conhecimento de Inteligência e comprometer os sistemas de informação do oponente (MINISTÉRIO DA DEFESA, 2014).

### 1.1.2 PROTEÇÃO DAS INFRAESTRUTURAS CRÍTICAS

A preocupação com a proteção das infraestruturas críticas tem crescido consideravelmente. Várias soluções e iniciativas têm sido desenvolvidas com o objetivo de tornar o ambiente preparado para lidar com atividades terroristas, espionagens, desastres naturais e situações de emergência.

Segundo Merabti et al. (2011), o avanço na modelagem de sistemas físicos e virtuais fornece ferramentas para maior entendimento e detecção de falhas. A partir disso, deriva-se a necessidade de desenvolver soluções para proteger os ativos de TI que suportam os sistemas críticos. Brown et al. (2006) defendem a utilização de diferentes técnicas de otimização para identificar vulnerabilidades e contribuir para tornar as infraestruturas mais resilientes a ataques físicos.

Em âmbito global, verifica-se a recente concepção e instalação de agências e organizações voltadas para este fim. Com isso, houve a elaboração e adoção de metodologias e políticas de preservação e gerenciamento de riscos, tais como EPCIP (União Europeia), NIPP (Estados Unidos), CPNI (Reino Unido) e CERT (Brasil). Estas fornecem conjuntos de boas práticas e ações para assegurar o funcionamento contínuo e resguardar a integridade física e operacional das instalações, estruturas e ativos nos setores de energia, comunicações, defesa e transportes contra os diferentes tipos de ameaças apresentados (YUSTA et al., 2011).

Em âmbito nacional, em julho de 2018, Itaipu e a União assinaram um acordo de parceria[1], com o Instituto Militar de Engenharia como interveniente, voltado para pesquisa,

[1] http://www.in.gov.br/materia/-/asset_publisher/KujrwOTZC2Mb/content/id/38934682/do3-2018-08-30-extrato-de-acordo-de-parceria-38934591

desenvolvimento e inovação do projeto "Centro de Estudos Avançados em Proteção de Estruturas Estratégicas - Fase 2: Consolidação", tendo cooperação técnica e financeira entre ITAIPU, Exército Brasileiro e o Parque Tecnológico de Itaipu do Brasil.

Com o avanço dos recursos tecnológicos, torna-se relevante o estudo de novas formas de incrementar a segurança e proteção das infraestruturas críticas. A dependência crescente de ferramentas de TI reforça a necessidade de que os sistemas devem estar aptos a lidar com vários tipos de ataques cibernéticos que tendem a explorar suas vulnerabilidades. Os principais problemas a serem superados envolvem segurança e integridade de dados, processos, instruções e informações sensíveis. Essas questões devem ser tratadas com prioridade devido ao potencial de impacto em sistemas críticos (YOUNIS et al., 2013).

O emprego de computação em nuvem vem ganhando cada vez maior importância e relevância por ser uma alternativa que apresenta vantagens como escalabilidade, baixo custo de manutenção e maior segurança (FAROOQI, 2016). A utilização de virtualizadores proporciona uma infraestrutura de computação capaz de criar ferramentas para construção e gerenciamento de nuvens públicas e privadas, tendo grande aplicação neste cenário. Nesse contexto, entre soluções disponíveis no mercado para o desenvolvimento de arquiteturas em nuvem, destaca-se o OpenStack, IaaS de código aberto. A implementação de sistemas por meio dessa ferramenta é mais simples pois contém serviços de computação, armazenamento e gerenciamento de imagens integrados. Além disso, esse *software* fornece ferramentas e painel de controle próprio, o que simplifica o processo de desenvolvimento das arquiteturas e manipulação das máquinas virtuais (BIST et al., 2013).

## 1.2 OBJETIVO

O presente projeto de fim de curso tem como objetivo a implementação de um ambiente computacional para simulação de ativos de tecnologia da informação e comunicação presentes em uma estrutura de infraestrutura crítica, por meio da virtualização dos componentes da arquitetura SCADA.

## 1.3 JUSTIFICATIVA

No contexto do acordo de parceria entre a União e Itaipu, é de grande importância e valor a realização de trabalhos acadêmicos no IME voltados à pesquisa, desenvolvimento e inovação orientados para a proteção de infraestruturas críticas. Assim, este projeto de fim de curso foi desenvolvido de forma a aprofundar o estudo de soluções que possam ser empregadas para este fim.

## 1.4 METODOLOGIA

O desenvolvimento do trabalho começou com a pesquisa bibliográfica e revisão dos conceitos fundamentais referentes à virtualização, computação em nuvem e infraestruturas críticas. Em seguida, foram estudadas as ferramentas necessárias para a implementação do ambiente de simulação. Com base nisso, foi definida uma arquitetura de referência, com camadas e componentes específicos. Concluida essa etapa, iniciou-se o desenvolvimento prático do projeto.

Após as definições iniciais, foi realizada a instalação do OpenStack no Laboratório de Informática do IME. Os computadores foram configurados com o Horizon e demais *frameworks* presentes na ferramenta de forma a viabilizar a criação de máquinas virtuais e construir um ambiente para simulação. Além disso, foi necessário alterar algumas configurações para permitir a conexão com a rede externa.

A partir disso, iniciou-se o processo de implementação e validação. Baseado na arquitetura SCADA, inicialmente foram configuradas as máquinas virtuais em nível operacional, com servidores *web*, de banco de dados e SCADA, atendendo às funcionalidades mínimas estudadas. Posteriormente, foram criadas máquinas de usuário para integração com as demais, além da configuração das políticas de segurança e *firewall*. Finalmente, a implementação foi concluída com a virtualização do simulador PLC para o OpenStack.

Em uma última etapa, foram realizados testes para verificar o funcionamento dos componentes e a integração entre eles. Também, foram conferidas as conexões entre as camadas, validando as permissões de acesso na rede.

## 1.5 ESTRUTURA

Este documento se estrutura em cinco capítulos, além de referências bibliográficas e apêndices desenvolvidos pelos autores. A primeira metade, composta pelos capítulos 2 e 3, apresenta uma fundamentação para o que foi desenvolvido. De maneira complementar, os capítulos 4, 5, 6 e 8 foram elaborados com base no desenvolvimento prático do projeto.

No capítulo 2, são abordados conceitos e embasamentos teóricos sobre infraestruturas críticas, sistemas de controle, computação em nuvem e virtualização. As ferramentas e *softwares* utilizados para o desenvolvimento do projeto (OpenStack, Rapid SCADA) tem seus recursos e funcionalidades analisadas no capítulo 3.

Com base no segundo e no terceiro, o quarto capítulo detalha a proposta de execução do projeto. São definidas e apresentadas a arquitetura base a ser seguida, as premissas e

considerações adotadas, bem como as etapas necessárias para sua realização e desenvolvimento.

No capítulo 5, é exposto o que foi desenvolvido e implementado com base na arquitetura de referência. As seções são divididas segundo as camadas e os componentes da arquitetura SCADA implementados. Além disso, o capítulo 6 apresenta o conteúdo e resultado dos testes e validações realizados para assegurar o correto funcionamento das máquinas virtuais, rede e recursos de segurança.

O desenvolvimento do trabalho é brevemente recapitulado no capítulo 7, no qual são feitas considerações e conclusões acerca do que foi realizado ao longo projeto. Também são enumeradas contribuições que o projeto apresenta para o desenvolvimento de pesquisas envolvendo virtualização de sistemas de infraestruturas críticas, concluindo a monografia.

O documento é finalizado com dois apêndices, nos quais são detalhadas etapas da implementação referentes à instalação do OpenStack e ao processo de criação das máquinas virtuais utilizando a ferramenta, servindo como referência para trabalhos futuros.

# 2 CONCEITOS FUNDAMENTAIS

Este capítulo destina-se a apresentar, explorar e analisar conceitos teóricos que serão empregados no desenvolvimento do projeto. Inicia cobrindo tópicos relacionados à infraestruturas críticas, ICS e finaliza abordando virtualização, computação em nuvem e a ferramenta OpenStack.

## 2.1 INFRAESTRUTURAS CRÍTICAS

A aplicação de metodologias para resguardar a integridade de sistemas de infraestrutura crítica é fundamental. No Brasil, são regulamentadas pela Política Nacional de Segurança de Infraestruturas Críticas[2], cujo objetivo é garantir a segurança, operação e prestação de serviços contínuas pelas infraestruturas críticas no território nacional. Nela são estabelecidos princípios para assegurar a continuidade de oferta, realizar gestão de ameaças e riscos, além de ações a serem aplicadas em eventuais interrupções de forma a minimizar os danos.

## 2.2 ICS

Sistemas de controle industriais são utilizados para controle e apoio a processos industriais, que visam preservar a disponibilidade e integridade das operações, além de gerenciá-las. As infraestruturas críticas beneficiam-se do emprego de ICS para assegurar sua contínua operabilidade, bem como atuar sobre eventuais falhas causadas devido à ações externas ao sistema.

Devido ao alto impacto resultante de interrupções ou alterações não programadas do fornecimento dos serviços, a segurança destes sistemas é essencial, logo devem ser desenvolvidos seguindo um conjunto de princípios, tais como disponibilidade, tolerância a falha, desempenho e segurança, de forma a assegurar seu bom funcionamento. É necessário garantir não apenas que os sistemas estejam sempre acessíveis aos usuários, mas também que apresentem robustês necessária para continuar operando minimamente, mesmo na hipótese de falhas. Maglaras et al. (2018) ainda reforçam a necessidade de estabelecer interconexões entre diferentes ICS como forma de estruturar políticas de ações de defesa mais ágeis e eficientes a ameaças cibernéticas.

[2] `http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/decreto/D9573.htm`

### 2.2.1 PLC

Controladores lógico programáveis são controladores utilizados em sistemas SCADA para monitorar componentes por todos os níveis hierárquicos do sistema e atuar através de um gerenciamento local sobre um equipamento. São componentes de *hardware* dedicados, projetados para comandar e monitorar máquinas ou processos industriais.

### 2.2.2 SCADA

Sistemas de supervisão e aquisição de dados são, de acordo com Daneels e Salter (1999), pacotes de software interfaceados com um componente físico por meio de PLCs aplicados a um nível de supervisor. Sistemas SCADA compõem os ICS e são empregados para controle e análise dos dados de equipamentos industriais. A monitoração de variáveis e estados desses componentes permite maior segurança e qualidade aos processos industriais, devido à constante adequação dos valores para os padrões estabelecidos e interrupções de eventuais anomalias e desvios.

A arquitetura SCADA é composta por servidores conectados a múltiplos controladores, cada qual com um conjunto próprio de parâmetros e variáveis de processos para avaliação, o que possibilita maior grau de escalabilidade. Estes são distribuídos pelas instalações, de forma que o sistema pode adquirir e transmitir dados e monitorá-los em um nó central.

O padrão ISA-95 (INTERNATIONAL SOCIETY OF AUTOMATION, 2019) estabelece que o desenvolvimento de sistemas SCADA siga uma arquitetura com quatro camadas, separando processos, atividades, sistemas e dispositivos. O diagrama da Figura 2.1 ilustra essa configuração. São definidas quatro camadas, que se comunicam por meio de protocolos de específicos, tais como DNP3, IEC 60870, OPC e MODBUS. Enquanto as camadas inferiores asseguram controle e supervisão, as superiores gerenciam as operações.

Na primeira, são situados os componentes responsáveis pela produção e controle dos processos, tais como PLCs e unidades de telemetria remota, que consolidam informações adquiridas e enviam para um comutador central. A segunda camada garante o monitoramento e supervisão, utilizando redes LAN para servidores SCADA e agentes supervisores por meio de HMIs, respectivamente (ENISA, 2016).

A terceira camada implementa o gerenciamento das operações, sendo composta por uma DMZ[3]. Esta atua de forma a reforçar a segurança do sistema, além de impedir acessos não autorizado tanto à rede da organização quanto aos componentes dos níveis

[3]DMZ é a região que promove isolamento entre a internet e a rede interna.

inferiores. Para isso, utiliza recursos de controle de acesso como *firewall* ou autenticação no *gateway*. Além disso, pode contar com sistemas de detecção de intrusões que gerenciam requisições externas, tráfego da rede e garantem que servidores e arquivos críticos não sejam manipulados (BOWEN et al., 2005). Finalmente, a quarta camada gerencia o acesso remoto aos recursos da organização através da internet.

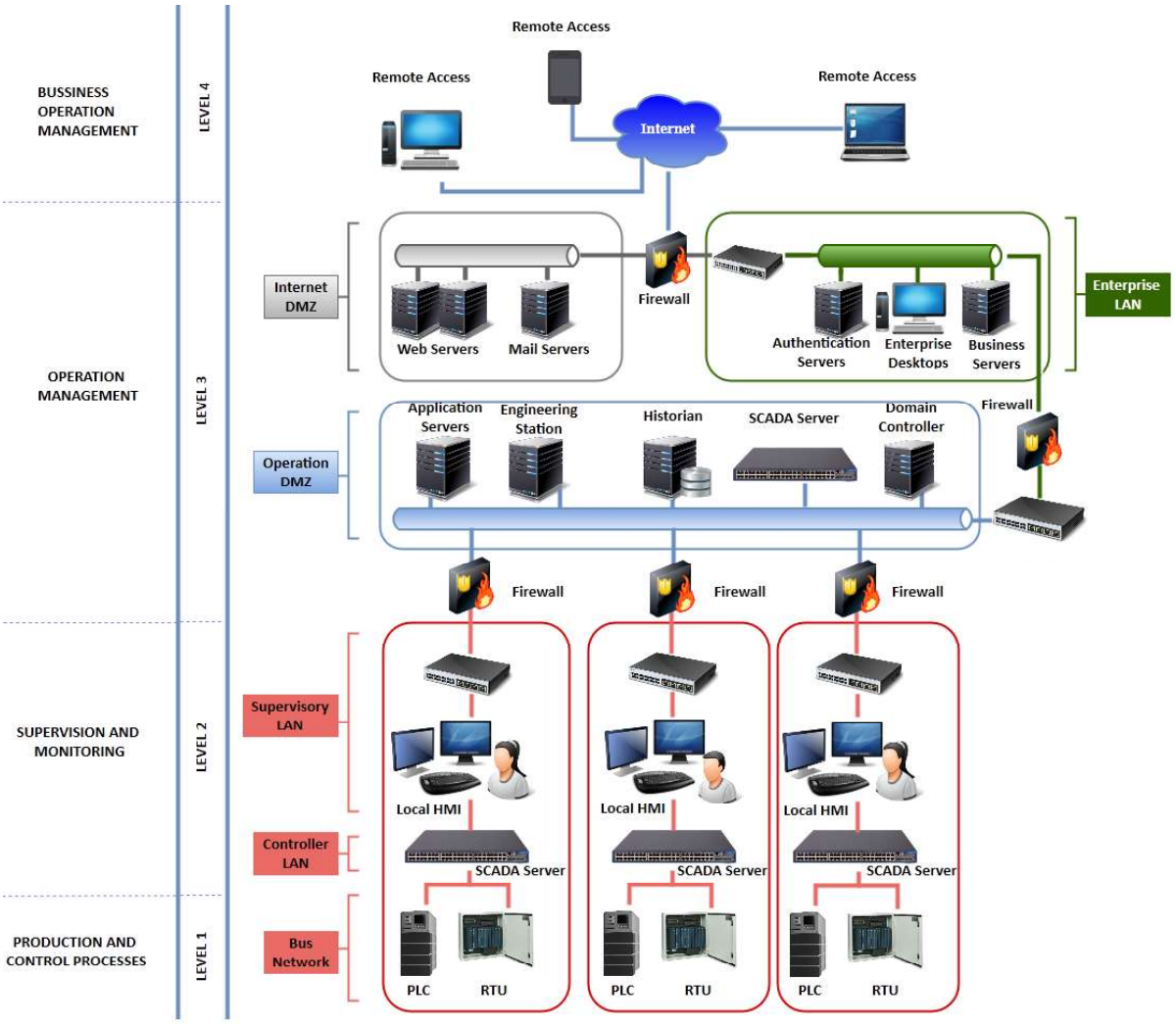



FIG. 2.1: Arquitetura SCADA com o padrão ISA-95 extraído de (ENISA, 2016).

## 2.3 VIRTUALIZAÇÃO

Virtualização é o processo de abstrair o hardware físico, substituindo-o por uma camada de software que comporta múltiplas instâncias de sistemas operacionais simultaneamente. Essa técnica permite gerenciar e alocar de forma eficiente os recursos do sistema físico que suporta o ambiente virtualizado, como CPU, memória, armazenamento e rede, entre as máquinas virtuais. Obtém-se portanto, maior flexibilidade e escalabilidade para o

desenvolvimento de arquiteturas (TANENBAUM; STEEN, 2006).

### 2.3.1 HIPERVISORES

No ambiente virtualizado, a interação entre as máquinas virtuais é realizado pelos hipervisores. Estes podem ser descritos como criadores e controladores das máquinas virtuais e dos recursos de hardware alocados. Em relação à implementação e função, podem ser divididos em hipervisores do tipo 1 e 2, como relacionado nos esquemas da Figura 2.2. Verifica-se que a interação com hardware e com as máquinas virtuais ocorre de maneira distinta. J. Popek e P. Goldberg (1974) definiram o hipervisor do tipo 1 como uma camada de software de monitoração de máquinas virtuais sobre uma camada de hardware, enquanto o tipo 2 requer uma camada intermediária de sistema operacional.

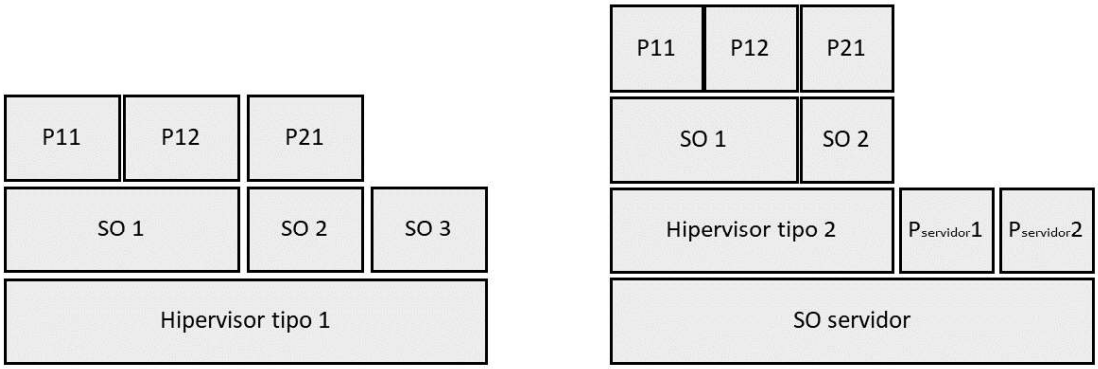



FIG. 2.2: Hipervisores dos tipos 1 e 2.

Os hipervisores do tipo 1, também chamados de *bare-metal*, são executados diretamente sobre o hardware físico do servidor, assumindo, assim, o papel de sistema operacional. Cada uma das múltiplas imagens que são executadas sobre este requer uma cópia do hardware físico. Dessa forma, o hipervisor funciona como gerenciador de cópias virtuais do hardware. Tal configuração resulta em maior eficiência e melhor desempenho das máquinas virtuais, que consomem hardware oferecido com maior disponibilidade.

Em contraste ao primeiro, os hipervisores do tipo 2 atuam sobre o sistema operacional do servidor, o que resulta em uma maior quantidade de recursos consumidos apenas para sua execução, quando comparados com o tipo 1. Isso ocorre pois os sistemas que executam o hipervisor hospedado não foram desenvolvidos com este fim. Consequentemente, nessa implementação há menor disponibilidade e desempenho, além do risco de que erros no sistema operacional afetem as máquinas virtuais. Entretanto, possibilita que outros pro-

cessos do usuário sejam executados no sistema operacional do servidor, simultaneamente às máquinas virtuais.

### 2.3.2 GÊMEO DIGITAL

No contexto de virtualização, o conceito de gêmeo digital refere-se ao uso de simulações para replicar virtualmente um sistema físico, seus componentes, características e interações. Permite que sejam obtidas várias informações relevantes referentes ao seu funcionamento, através de monitoração, testes de dispositivos, análises de segurança e aprendizado e predição de comportamento. Essa abordagem demonstra-se de grande utilidade para sistemas de infraestrutura crítica e ICS.

A aquisição de dados referentes à operação do sistema físico pode ocorrer no ambiente digital por meio de capturas do estado dos componentes replicados. Isso permite obter uma maior quantidade de informação em um intervalo de tempo menor do que seria possível com o primeiro, variando de acordo com a capacidade de processamento da estrutura em que a simulação está sendo executada (UHLEMANN et al., 2017). Um monitoramento contínuo deste fornecerá métricas que podem ser validadas com o ambiente físico e aplicadas para predizer comportamentos futuros.

Ademais, os dispositivos do sistema físico podem ser testados individualmente, de forma integrada com o gêmeo digital. Utilizando as métricas e dados previamente obtidos com a etapa de aquisição, pode-se verificar se os componentes estão operando conforme esperado. Torna-se possível, portanto, uma análise mais rápida da integração das partes do sistema, além da identificação de erros e falhas.

O uso de uma cópia digital da arquitetura também permite que sejam realizados testes e análises de segurança no sistema, porém sem a exposição à riscos de comprometer a estrutura física real. Esta técnica pode ser empregada para simular ataques e analisar as alterações e comportamentos anômalos causados nos controladores físicos, possibilitando a posterior mitigação de vulnerabilidades (ECKHART; EKELHART, 2018). Tais medidas contribuem para a melhor definição de uma política de segurança.

## 2.4 COMPUTAÇÃO EM NUVEM

Computação em nuvem é um conjunto de abordagens, ferramentas e tecnologias que visa oferecer serviços sob demanda de maneira remota, isto é, sem necessidade de unidades de computação física com elevada capacidade de hardware. Estas deixam de ser provedoras dos serviços e passam a ser usadas apenas para acessar a nuvem e consumir os

recursos disponíveis. Os serviços disponibilizados variam de acordo com a abordagem empregada na nuvem, e podem englobar gerenciamento, automação e alocação de recursos de rede, armazenamento, virtualização, sistema operacional, *middleware*, dados e aplicações, implementados por componentes de software que se comunicam através de trocas de mensagens.

As arquiteturas das nuvens são desenvolvidas de forma a entregar os serviços de maneira eficiente, com alocação dinâmica dos recursos, escalável e transparente ao usuário. Isso possibilita a utilização destas para o desenvolvimento de estruturas complexas e personalizadas que demandem capacidade computacional e recursos de armazenamento e rede variável. Apesar das vantagens destacadas, o acesso à nuvem pode ser comprometido devido à falha ou indisponibilidade de acesso à rede.

Os serviços da computação em nuvem podem ser implementados, principalmente, nas seguintes arquiteturas: infraestrutura como serviço (IaaS), plataforma como serviço (PaaS) e software como serviço (SaaS). O esquema da Figura 2.3 representa estes três tipos em relação aos recursos e dependências oferecidos. Na base da pirâmide estão localizados serviços básicos como rede e virtualização, que são utilizados em todas as soluções, enquanto dados e aplicações estão no seu topo como serviços específicos que consomem dos demais. As subseções subsequentes exploram e detalham as características destes três modelos de implementação de nuvem.

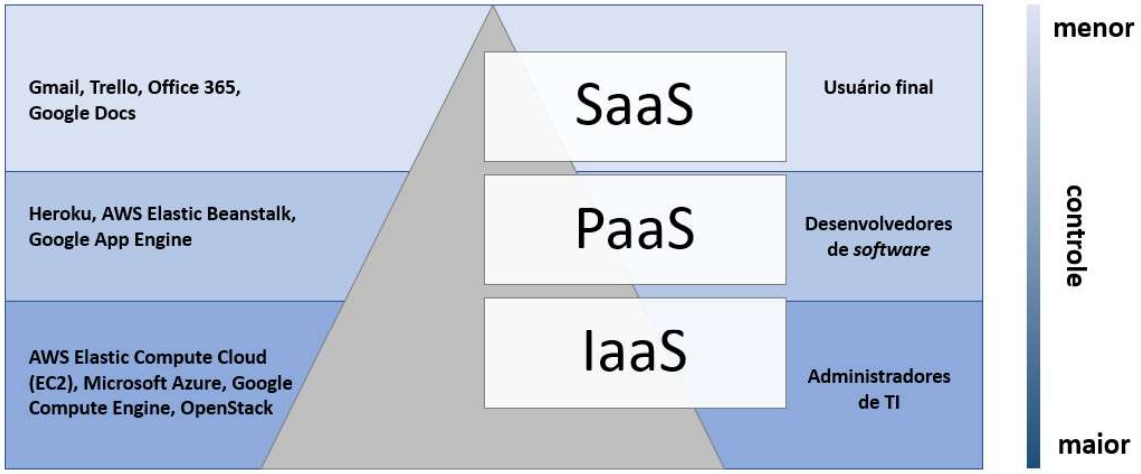



FIG. 2.3: Tipos de computação em nuvem, adaptado de Adeeb (2014).

### 2.4.1 IAAS

Os serviços de infraestrutura em nuvem são modelos responsáveis por fornecer recursos de infraestrutura necessários para as camadas superiores, provendo e gerenciando virtualização, servidores, discos rígidos, armazenamento e rede. Os provedores de IaaS podem também oferecem bancos de dados, filas de mensagens e outros serviços acima da camada de virtualização. Além disso, através dessa arquitetura, é possível alocar os recursos de hardware sob demanda, de forma a atender as necessidades do sistema operacional que está sendo executado.

O consumo deste tipo de serviço isenta o usuário da necessidade de possuir hardware físico, bem como de configurá-lo e realizar manutenções. Nessas plataformas, estes são responsáveis apenas pela gestão e configuração de aplicativos, dados, tempo de execução, middleware e sistemas operacionais.

### 2.4.2 PAAS

Plataforma como serviço é usado por aplicativos e outros desenvolvimentos ao mesmo tempo que fornecem aplicativos de nuvem para o software. Com o PaaS, os desenvolvedores têm uma estrutura onde é possível desenvolver ou personalizar aplicativos tornando o desenvolvimento, teste e implantação de aplicativos mais rápidos, simples e econômicos.

A PaaS permite criar aplicativos usando componentes de software integrados ao middleware. Aplicativos que usam PaaS herdam caracterísitcas de nuvem como escalabilidade, disponibilidade, multilocação, habilitação SaaS, entre outras. Com sua utilização, ocorre uma redução na quantidade de codificação necessária, automatização de processos e auxiliam a migrar aplicativos para o modelo híbrido.

### 2.4.3 SAAS

Software como serviço fornece toda a estrutura desde gerenciamento de redes até dados e aplicações.

Por essa razão, é o tipo de serviço mais acessível para o usuário final, sendo amplamente utilizado devido a grande quantidade de APIs criadas para aproveitá-los. Esses serviços são implementados de forma que o usuário o acessa diretamente pela web, não sendo necessário downloads ou instalações.

Os principais tipos de SaaS incluem e-mail, aplicações de colaboração e gerenciamento de relacionamento com cliente. Muitas grandes empresas, que não são fornecedoras de

software, utilizam o SaaS como uma fonte adicional de receita a fim de se tornar mais competitiva no mercado.

A Tabela 2.1 resume o gerenciamento de recursos em cada um dos tipos de implementação de computação em nuvem analisados. A escolha da abordagem depende dos requisitos do usuário, não havendo um tipo melhor. Neste projeto, será utilizado o OpenStack, cujas características são expostas na próxima seção, componente de software implementado como IaaS.

TAB. 2.1: Recursos gerenciados pela nuvem, adaptado de Goran et al. (2014)

| | **IaaS** | **PaaS** | **SaaS** |
|---|---|---|---|
| Aplicações | | | |
| Dados | | | |
| Runtime | | | |
| Middleware | | | |
| SO | | | |
| Virtualização | | | |
| Servidores | | | |
| Armazenamento | | | |
| Rede | | | |

# 3 FERRAMENTAS E SOFTWARES

Este capítulo destina-se a apresentar as ferramentas e *softwares* utilizados para o desenvolvimento deste projeto, bem como explorar as funcionalidades fornecidas que foram necessárias para sua execução.

## 3.1 OPENSTACK

OpenStack é uma plataforma de software de código aberto implementada como IaaS, através de componentes que controlam o acesso aos seus recursos. Foi inicialmente desenvolvida em conjunto pela NASA e a Rackspace para proporcionar uma infraestrutura de computação e armazenamento em nuvem. Hoje é regulado pela OpenStack Foundation[4], contando com uma comunidade de mais de 600 empresas.

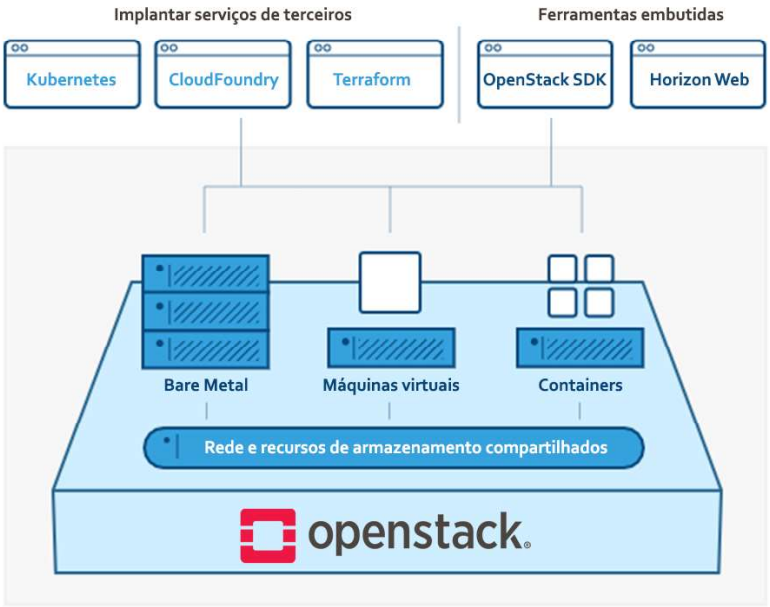



FIG. 3.1: Diagrama de recursos do OpenStack.

A arquitetura do software é estruturada em projetos que podem ser conectados de acordo com a demanda da aplicação. Uma implementação OpenStack contém serviços de computação, redes e armazenamento, conforme ilustrado no diagrama da Figura 3.1. O controle da plataforma é realizado tanto por meio de linha de comando quanto pelo

[4]https://www.openstack.org/foundation/

painel de controle web - componente Horizon, abordado como tópico específico adiante nesta seção. Além disso, permite a implantação de serviços de terceiros.

Os componentes do OpenStack podem ser instalados de maneira independente e se comunicam por meio de protocolos de troca de mensagens. A segurança baseia-se em políticas de autenticação e autorização, implementadas pelo projeto Keystone. O gerenciamento de rede, imagens e máquinas virtuais á realizado por meio de APIs específicas. Wen et al. (2012) destacam o suporte ao uso de diferentes hipervisores, além da estrutura modularizada em projetos autônomos, como características diferenciais do OpenStack que o tornam mais flexível e escalável. O diagrama da Figura 3.2 sintetiza a arquitetura, na qual o provedor de identidade promove a integração dos projetos.

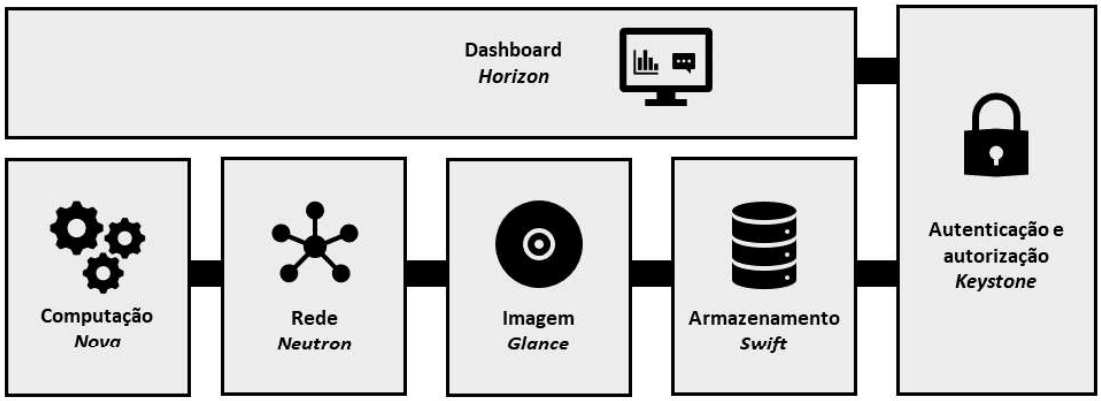



FIG. 3.2: Arquitetura dos projetos do OpenStack.

As próximas subseções abordam de maneira mais detalhada as características e recursos dos principais componentes do OpenStack: Keystone, Horizon, Nova, Neutron, Glance e Swift.

### 3.1.1 KEYSTONE

Keystone (*OpenStack Identity*) é o projeto responsável pela identificação, autenticação e autorização no OpenStack, gerenciando os relacionamentos de usuários, permissões e papéis. Cada usuário possui um conjunto de serviços e *endpoints* que pode acessar, o que é fornecido aos demais projetos como um diretório central de registros de autorização. A autenticação pode ser realizada tanto por meio de par usuário-senha quanto por meio de *tokens*, sendo consumida por todos os serviços da plataforma.

### 3.1.2 HORIZON

Acessar, configurar e gerenciar componentes da nuvem por meio da linha de comando podem ser atividades custosas. O Horizon (*OpenStack Dashboard*) resolve este problema fornecendo uma interface gráfica web para os usuários da plataforma, possibilitando de maneira mais simples o gerenciamento de outros projetos e serviços, contribuindo com a escalabilidade do desenvolvimento.

### 3.1.3 NOVA

Uma implementação do OpenStack é composta por máquinas virtuais que são executadas pelo Nova (*OpenStack Compute*). Este componente é responsável pelo gerenciamento dos recursos de computação e processamento alocados nos demais componentes da plataforma.

As instâncias das máquinas virtuais são gerenciadas pelo hipervisor, que comunica-se com o Nova por meio de chamadas à API. Quando um usuário faz uma requisição para criação de máquinas virtuais, o Nova a processa e redireciona ao hipervisor outra requisição que este consiga interpretar e processar.

### 3.1.4 NEUTRON

O gerenciamento de redes na plataforma é implementada pelo Neutron (*OpenStack Networking*). A rede se estrutura em gerencial (comunicação entre os componentes do OpenStack, acessível apenas na plataforma), externa (possibilitar acesso à internet às máquinas virtuais) e para APIs, com um domínio para acesso aos serviços do OpenStack. O componente também oferece recursos que podem ser empregados para definição de políticas de segurança.

### 3.1.5 GLANCE

Glance (*OpenStack Image Service*) é o componente de gerenciamento das imagens das máquinas virtuais usadas pelo Nova. Através deste módulo, é possível consultar, registrar e recuperar imagens, que podem ser armazenadas de diferentes maneiras. O controle das imagens é realizado por meio de requisições a API do serviço, o que fornece transparência aos usuários.

Para criação das instâncias no OpenStack, são utilizadas imagens anteriormente adicionadas ao projeto. Esse processo utiliza arquivos que contenham um disco virtual com sistema operacional instalado.

Outro recurso fornecido pelo componente é a capacidade de criar *snapshots*[5] das máquinas virtuais. A partir desses, novas máquinas virtuais podem ser iniciadas utilizando-os como imagem. Isso possibilita replicar e migrar instâncias entre diferente projetos do OpenStack, preservando o estado, instalações e configurações já realizados na máquina.

### 3.1.6 SWIFT

O Swift (*OpenStack Object Storage*) é utilizado para gerenciar o armazenamento dos dados na plataforma. O componente foi desenvolvido de maneira distribuída, de forma a oferecer um serviço que possibilite escalabilidade e durabilidade, suportando diferentes formatos de arquivos de maneira eficiente e segura (OPENSTACK, 2019).

## 3.2 RAPID SCADA

O Rapid SCADA é um *software* SCADA de código aberto e gratuito, desenvolvido para construção de sistemas automatizados. Pode ser aplicado em ambientes que contenham controladores, sensores e atuadores, tais como sistemas industriais, sistemas de automação, dispositivos de segurança e controle de acesso. O aplicativo está definido por bibliotecas e pacotes, o que permite adaptar às necessidades específicas do uso.

O funcionamento do *software* ocorre por meio da coleta de dados provenientes de controladores, processamento e fornecimento de informações para um componente atuador. Essa comunicação é realizada utilizando protocolos padronizados, como OPC e MODBUS, permitindo integração com uma diversidade de dispositivos (RAPID SCADA, 2019).

### 3.2.1 RECURSOS

A arquitetura do Rapid SCADA é baseada em um servidor com três níveis de aplicação. A comunicação com o usuário é feita utilizando uma estação *web*, enquanto que com os equipamentos industriais ocorre por meio do comunicador. Intermediando estes dois níveis, há uma camada de gerenciamento e processamento de dados. Tal estrutura está representada no diagrama da Figura 3.3.

A *webstation* é uma aplicação *web* que possibilita a exibição das informações processadas para os usuários finais, por meio de tabelas, diagramas e relatórios. Também possibilita o envio de comandos para controle do sistema.

O nível de servidor é responsável por todo o processamento lógico dos dados adquiridos pelo comunicador, realizando cálculos e gerenciando os arquivos de dados obtidos. A

[5]Registro instantâneo da imagem e do estado da máquina virtual.

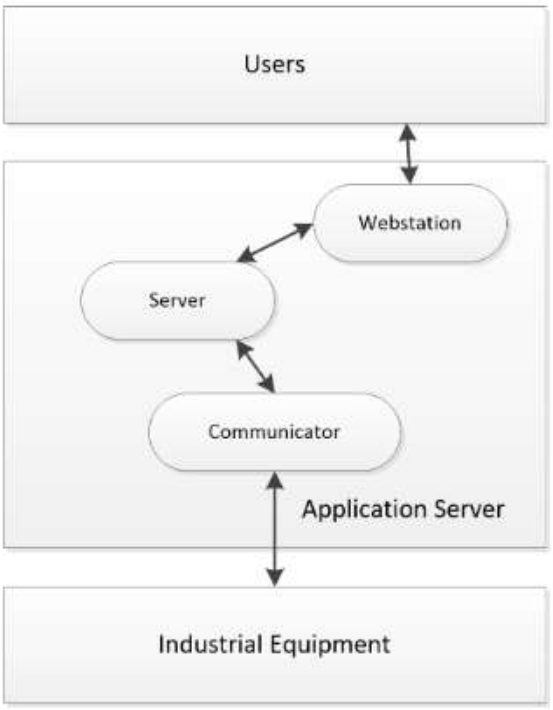



FIG. 3.3: Arquitetura do Rapid SCADA, extraído de Rapid SCADA (2019).

operação dessa camada ocorre de maneira contínua, independente de haver ou não algum acesso de usuário. Além disso, não possui interface para o usuário, sendo seu controle apenas disponível para o administrador.

O comunicador atua sobre os controladores de duas formas. A primeira é definida pelo processo de extração de informações do estado atual do controlador, geração de arquivos e envio para a aplicação de servidor. A segunda está relacionada com a tradução de uma instrução recebida do servidor, enviando um comando para o equipamento.

# 4 PROPOSTA

Conforme exposto no capítulo anterior, torna-se cada vez mais relevante o estudo de alternativas para a proteção dos ativos de tecnologia da informação e comunicações no contexto de infraestruturas críticas. Neste trabalho, propoe-se estruturar um ambiente virtualizado capaz de simular tais instalações, seus componentes e recursos.

Este capítulo destina-se a apresentar a proposta de implementação e desenvolvimento prático do projeto de fim de curso. Estão expostas as premissas e considerações que foram adotadas, de forma a delimitar o escopo do trabalho. Com isso, almeja-se atingir o objetivo de estruturar um ambiente computacional para infraestruturas críticas.

Para isso, são definidas as ferramentas necessárias e suas respectivas aplicações. Conjuntamente, é apresentada a arquitetura implementada no projeto, que foi baseada na ENISA (2016) cuja ilustração foi representada na Figura 2.1.

Nesse contexto, são abordadas as camadas e regiões que a compõem. Para estas, também é descrito o funcionamento esperado de cada componente virtualizado. Ademais, propõe-se uma solução para virtualizar as redes, as políticas de segurança e o controle de tráfego no ambiente de simulação.

## 4.1 AMBIENTE

Como ferramenta principal para a estruturação do ambiente computacional para infraestruturas críticas, propõe-se o uso do OpenStack, cujas características estão listadas no capítulo anterior. A principal funcionalidade utilizada nesse projeto é o *dashboard* Horizon, *framework* que possibilita maior controle sobre o gerenciamento dos recursos virtualizados, além de contribuir para a redução do tempo necessário para implementação, conferindo maior agilidade ao processo de criação e configuração de máquinas virtuais.

## 4.2 ARQUITETURA

O escopo da implementação do projeto foi limitado por meio da seleção de parte dos elementos expostos na Figura 2.1. Esta escolha foi realizada consolidando componentes com aplicações semelhantes em uma única máquina virtual. Assim, foi obtida uma arquitetura resumida que atende aos objetivos do projeto de estruturar um ambiente computacional para infraestruturas críticas.

Considera-se que tais modificações não descaracterizam a arquitetura SCADA. Isso porque, uma vez implementada, pode ser expandida utilizando os recursos do OpenStack de cópia e replicação de máquinas virtuais. Dessa forma, as configurações comuns às máquinas não precisam ser realizadas novamente, tornando o processo escalável.

A partir da arquitetura de referência, foi projetada a arquitetura proposta, ilustrada na Figura 4.1. Verifica-se que foi preservada a estrutura de quatro camadas, assim como a arquitetura de redes, dividida em *Internet-DMZ*, *Enterprise LAN*, *Operation-DMZ* e *Supervisory LAN*. Entretanto, a quantidade de componentes foi reduzida, sendo as alterações descritas nos parágrafos subsequentes.

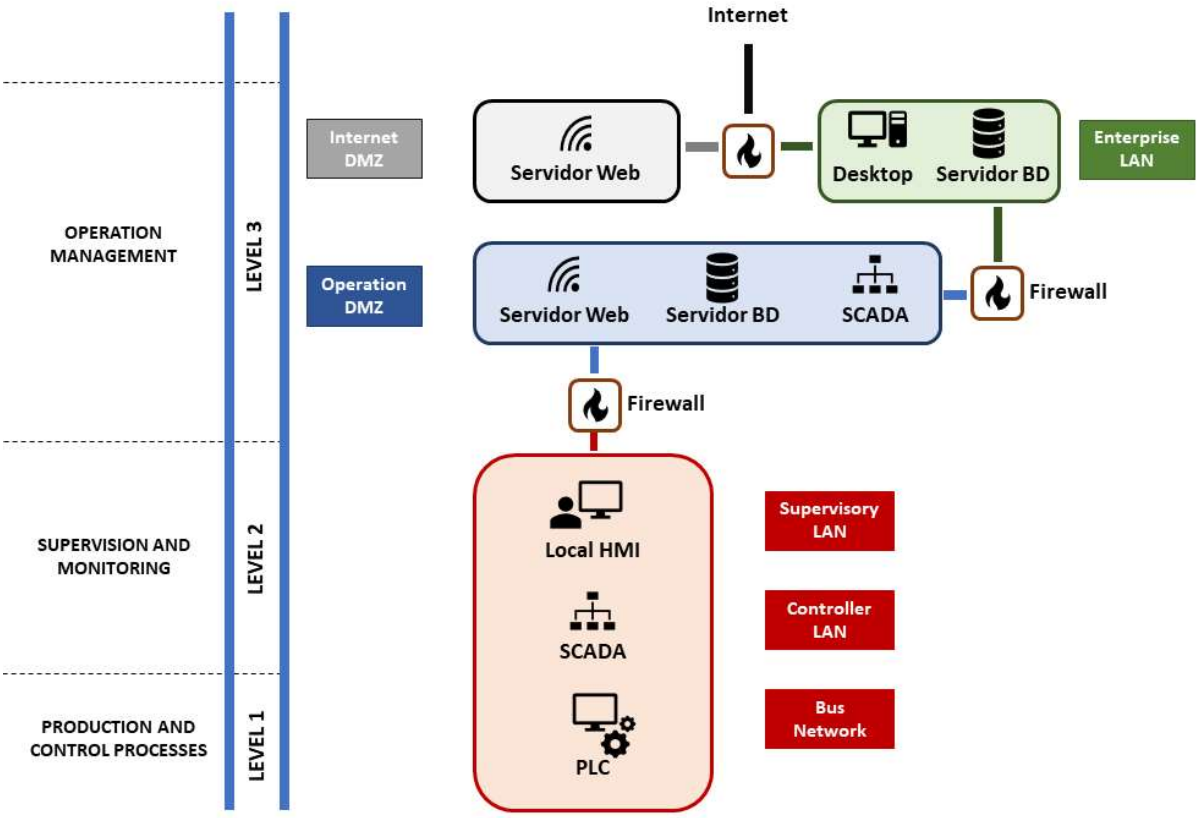



FIG. 4.1: Arquitetura proposta para implementação.

A região *Internet-DMZ* inicialmente era composta por dois servidores: *web* e e-mail. Na arquitetura proposta, o segundo foi suprimido. Considerando a finalidade de verificar o funcionamento da DMZ como região de isolamento, considera-se que é suficiente apenas o uso do servidor *web* como recurso visível tanto para a internet quanto para a rede *Enterprise LAN* nessa região.

Na *Enterprise LAN* os servidores foram consolidados em um servidor de banco de dados. A inclusão de servidores de autenticação e de arquivos pode ser realizada con-

forme necessidade utilizando o mesmo princípio, sendo desconsiderada nesse projeto para simplificar a arquitetura. Além diso, está mantida a figura do *desktop*, que deve ser capaz de realizar requisições tanto ao servidor de banco de dados quanto ao servidor *web* da *Internet-DMZ*.

De maneira similar, na arquitetura proposta a *Operation-DMZ* é composta por três servidores: um *web*, um de banco de dados, além de SCADA. Por fim, as regiões de supervisão e controle são mantidas com uma máquina de usuário, um servidor SCADA e um PLC.

## 4.3 VIRTUALIZAÇÃO

A virtualização do ambiente foi realizada utilizando o OpenStack. As máquinas exibidas na arquitetura proposta são implementadas por meio de máquinas virtuais específicas, utilizando sistemas operacionais distintos, conforme a sub-seção subsequente. As redes e componentes de *firewall* são virtualizados por meio de recursos específicos da ferramenta.

### 4.3.1 MÁQUINAS

Para virtualização dos servidores, as máquinas virtuais criadas empregam o sistema operacional Ubuntu 16.04.6 LTS (Xenial Xerus). Este foi escolhido por ser um *software* gratuito e atender às necessidades mínimas para cada um dos componentes. Em contraste, as máquinas de usuário da *Local HMI* executam a versão de avaliação do sistema operacional Windows Server 2012 R2 fornecida pelo Cloudbase Solutions[6]. Essa distribuição foi escolhida por estar pronta para uso no OpenStack, não sendo necessária nenhuma conversão de formatos. Especificamente, tanto a distribuição do Ubuntu quanto do Windows foram selecionadas por serem a versões constantes no guia de imagens[7] do OpenStack.

A virtualização dos servidores SCADA ocorre por meio do Rapid SCADA, instalado em uma máquina servidora conforme descrito anteriormente. O PLC foi virtualizado em uma máquina com Ubuntu. Essa relação de máquinas, regiões e sistemas operacionais é sumarizada na Tabela 4.1 a seguir.

### 4.3.2 REDES E FIREWALL

No ambiente de simulação, a arquitetura de redes do sistema proposto é composta de duas redes, uma pública e uma privada. A segunda está dividida em quatro sub-redes,

[6] `https://cloudbase.it/windows-cloud-images/`

[7] `https://docs.openstack.org/image-guide/obtain-images.html`

TAB. 4.1: Máquinas virtuais na arquitetura proposta.

| **Componente** | **Sistema operacional** |
|---|---|
| Servidor *Web* | Ubuntu |
| Servidor BD | Ubuntu |
| *Desktop* | Ubuntu |
| Local HMI | Windows |
| SCADA | Ubuntu |
| PLC | Ubuntu |

conforme as regiões da figura de referência. São elas *Internet-DMZ*, *Enterprise LAN* e *Operation-DMZ*, além de uma quarta que consolida *Supervisory LAN*, *Controller LAN* e *BUS Network*. Essa divisão foi realizada para agrupar os componentes que estão sujeitos às mesmas regras de segurança e restrição de tráfego.

Os componentes de *firewall* presentes entre as regiões estão implementados utilizando políticas de controle de tráfego de rede. Dessa forma, não há instâncias de máquinas virtuais específicas para este fim, em contraste com o que é apresentado na Figura 2.1. Essa escolha deve-se à possibilidade de utilizar os recursos do OpenStack, que permitem a definição dos grupos de segurança de ingresso e egresso, para suprir essa necessidade. Nota-se, entretanto, que essa solução poderia ser incrementada utilizando outros recursos externos, que não os fornecidos pelo OpenStack. Para limitar o escopo, o projeto seguiu apenas com a utilização dessa funcionalidade.

# 5 IMPLEMENTAÇÃO DO SIMULADOR DE INFRAESTRUTURA CRÍTICA

A implementação do simulador de infraestrutura crítica seguiu os tópicos abordados no capítulo anterior. As atividades realizadas durante este projeto encontram-se descritas nas seções que compõem este capítulo. Dessa forma, são abordadas as camadas da arquitetura exposta na Figura 4.1, além das soluções empregadas para segurança e arquitetura de redes.

## 5.1 CONSIDERAÇÕES INICIAIS

O objetivo deste trabalho é contruir um ambiente computacional em nuvem baseado na arquitetura SCADA apresentada na Figura 2.1. Com isso, busca-se oferecer uma estrutura para teste e desenvolvimento de ferramentas de segurança voltadas para sistemas de infraestrutura crítica.

Para isso, os componentes de cada camada da arquitetura proposta na Figura 4.1 foram virtualizados, sendo o OpenStack utilizado como ferramenta para configuração e gestão de máquinas virtuais, as quais representam os componentes físicos do sistema real.

A implementação ocorreu utilizando a interface gráfica do painel de controle Horizon. Por meio deste, é possível realizar a configuração de redes, grupos e políticas de segurança, gerenciamento de imagens e instâncias de máquinas virtuais sem a necessidade do uso de instruções por linha de comando. A versão da plataforma empregada é 16.0.0 e foi instalada e validada conforme os passos descritos no APÊNDICE 1 deste texto. Ademais, o processo de criação de máquinas virtuais seguiu os passos descritos no APÊNDICE 2.

O desenvolvimento do projeto segue a arquitetura de camadas apresentada na fundamentação teórica, na Figura 2.1. Foi adotada uma metodologia *top-down*, iniciando pelos componentes das camadas superiores. As subseções subsequentes apresentam as configurações e testes realizados para implementação de cada módulo.

Ressalta-se que a arquitetura implementada é uma simplificação de um modelo mais complexo, que pode ser ampliada e adaptada conforme necessidade. O enfoque do trabalho está nas camadas, regiões e componentes gerais comuns a sistemas de infraestruturas críticas.

Destaca-se, ainda, que devido a finalidade de simulação e com intuito de facilitar o

desenvolvimento, foi utilizado apenas o protocolo de redes IPv4. Tal escolha simplifica o processo de implementação e configuração, possibilitando maior enfoque no simulador SCADA para infraestruturas críticas.

## 5.2 CAMADA DE GERENCIAMENTO DE OPERAÇÕES

A simulação desta camada foi realizada por meio da virtualização de rede e máquinas divididas conforme arquitetura proposta na Figura 4.1. Ela está segregada nas regiões *Internet-DMZ*, *Entrerprise LAN* e *Operation-DMZ*, cada uma com seus componentes específicos, conforme sumarizado na Tabela 5.1. Os componentes básicos foram configurados e posteriormente replicados por meio de *snapshots*. As configurações específicas foram realizadas posteriormente.

TAB. 5.1: Componentes da camada de gerenciamento de operações.

| **Região** | ***Web*** | **Banco de dados** | ***Desktop*** | **SCADA** |
|---|---|---|---|---|
| *Internet-DMZ* | x | | | |
| *Entrerprise LAN* | | x | x | |
| *Operation-DMZ* | x | x | | x |

As subseções subsequentes apresentam as definições e características de cada um dos componentes.

### 5.2.1 SERVIDOR WEB

Este módulo objetiva fornecer uma máquina virtual que execute um servidor *web* e possa ser empregada na aquitetura do simulador. Na camada de gerenciamento de operações, este componente deve ser replicado duas vezes, nas regiões *Internet-DMZ* e *Operation-DMZ*.

A criação deste componente foi realizada a partir de uma imagem do sistema operacional Ubuntu 16.04.6, na qual foi realizada a instalação do servidor HTTP Apache e PHP. A Figura 5.1 lista as especificações das aplicações extraídas do console da máquina virtual. Após a finalização das instalações, foi salva uma *snapshot* da imagem da máquina virtual, de forma a replicá-la posteriormente. A partir dessa máquina, torna-se apenas necessário realizar a atualização do endereço IP do servidor de banco de dados e alterar as consultas na página PHP.

A primeira instância do servidor *web*, na região *Internet-DMZ*, deve viabilizar uma interface de acesso disponível tanto para conexões internas quanto externas à sub-rede

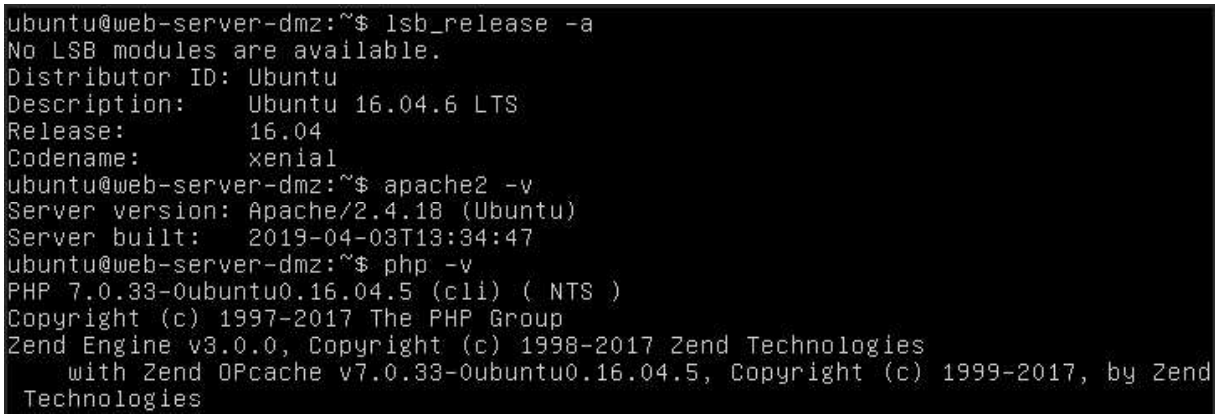



FIG. 5.1: Configurações do servidor *web*.

*Internet-DMZ*. Também, deve conectar-se a um servidor de banco de dados presente na região *Enterprise LAN*, no qual são realizadas consultas e inserções de dados referentes aos eventos de acesso ao servidor.

Semelhante ao antecessor, o servidor na região *Operation-DMZ* possibilita o registro de *logs* de acesso, porém difere em dois aspectos. As requisições podem ocorrer tanto da própria região, quanto a partir de máquinas de interface da *Supervisory LAN*. Outro ponto de diferença é o servidor de banco de dados acessado, que pertence à mesma região. Como o objetivo deste componente é viabilizar a ligação entre as duas regiões, o padrão adotado nessa segunda instância será semelhante à da primeira.

### 5.2.2 SERVIDOR DE BANCO DE DADOS

Conforme descrito na proposta, na camada de gerenciamento de operações são definidos dois servidores de banco de dados - os mesmos relacionados na sub-seção anterior. O objetivo destes é possibilitar, de forma genérica, o registro de dados provenientes de outras máquinas.

Seguindo o padrão do módulo anterior, este foi executado criando uma máquina virtual com sistema operacional Ubuntu 16.04.6, na qual foi instalado e configurado o servidor de banco de dados. Visando a atender essa necessidade e os recursos do sistema, foi escolhido o MySQL Server, cuja versão utilizada é exibida na Figura 5.2. Após completar as instalações e configurações, a imagem da máquina foi salva para ser reutilizada.

No contexto da aplicação, cada servidor de banco de dados contém uma tabela de eventos, ilustrada na Figura 5.3, para registrar quando os acessos ao respectivo servidor *web* ocorrem. Tanto a consulta quanto a inserção desses dados são realizadas pela página PHP presente no servidor *web* no momento de acesso à mesma.

```
ubuntu@enterprise-db-server:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 16.04.6 LTS
Release:        16.04
Codename:       xenial
ubuntu@enterprise-db-server:~$ mysql -V
mysql  Ver 14.14 Distrib 5.7.27, for Linux (x86_64) using  EditLine wrapper
```

FIG. 5.2: Configurações do servidor de banco de dados.

```
mysql> use logs
Database changed
mysql> select * from events;
+----------+---------------------+
| event_id | event_date          |
+----------+---------------------+
|        1 | 2019-08-05 03:29:04 |
|        2 | 2019-08-05 15:29:25 |
|        3 | 2019-08-05 15:29:45 |
+----------+---------------------+
3 rows in set (0.00 sec)
```

FIG. 5.3: Tabela de registro de eventos de acesso.

### 5.2.3 DESKTOP

Além do servidor de banco de dados, a *Enterprise LAN* contém também máquinas *desktop*, representando máquinas de usuários do sistema na rede corporativa. Dessa forma, sem perda de generalidade, foi criada uma máquina virtual com Ubuntu 16.04.6. Outras máquinas, com sistema operacional distinto, podem ser criadas nessa região. Entretanto, considera-se que apenas uma é suficiente para exemplificar a arquitetura.

O objetivo deste componente limitou-se a validar os níveis de permissão de acesso. Dessa forma, a partir dessa máquina deve ser possível acessar o servidor *web* da *Internet-DMZ* e o servidor de banco de dados, o que foi verificado.

## 5.3 CAMADA DE SUPERVISÃO E MONITORAMENTO

### 5.3.1 LOCAL HMI

Este módulo foi desenvolvido utilizando uma máquina virtual criada a partir de uma imagem do sistema operacional Windows Server 2012 R2, conforme descrição na proposta do projeto.

As máquinas de usuário da camada de supervisão são responsáveis por possibilitar que os operadores do sistema acessem os servidores SCADA pela *webstation*. Serve, portanto, como a interface na qual o usuário interage com o sistema. Além disso, também tem acesso ao servidor *web* criado na sub-rede *Operation-DMZ*.

### 5.3.2 SERVIDOR SCADA

Na arquitetura proposta da Figura 4.1, o servidor SCADA deve A criação do servidor SCADA foi realizada em uma máquina originada a partir de uma *snapshot* do servidor *web*. Nesta instância, baseada na documentação do Rapid SCADA (2019), foi realizada a instalação do *software*.

Em seguida, foi realizada a configuração do Rapid SCADA em níveis de dispositivos, linhas de comunicação e canais de entrada e saída, de forma a realizar a integração com o PLC. Nessa etapa, são definidos os parâmetros da comunicação, tais como protocolo utilizado, tipo de comando a ser enviado e ação a ser realizada.

Nesse projeto, o servidor SCADA deve viabilizar a comunicação com o controlador lógico programável. Para isso, foi empregado o protocolo MODBUS/TCP, o qual utiliza a porta 502. Além disso, foi selecionada a função 03 (leitura de registros retentivos) para requisição de saídas do PLC. As Figuras 5.4 e 5.5 apresentam os arquivos de configuração utilizados para as definições de canais e linhas de comunicação e interação com os dispositivos, respectivamente.

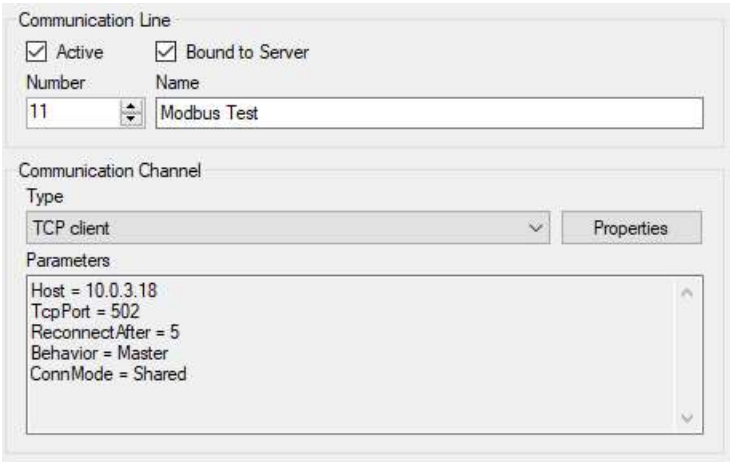



FIG. 5.4: Configuração dos canais de comunicação utilizando o Rapid SCADA.

Finalmente, após o término das configurações, os arquivos criados foram carregados no servidor SCADA. Após a reinicialização do serviço, o *software* Rapid SCADA pode ser acessado de maneira remota pela *webstation*, o que possibilita a utilização do servidor pelos operadores do sistema por meio das máquinas de usuário.

A partir da página *web*, é possível realizar a monitoração das linhas de comunicação e dispositivos configurados anteriormente, com leitura e envio de dados. A Figura 5.6 apresenta a interface gráfica do módulo *web* do servidor SCADA, acessado por uma máquina de usuário presente na camada de supervisão e monitoramento.

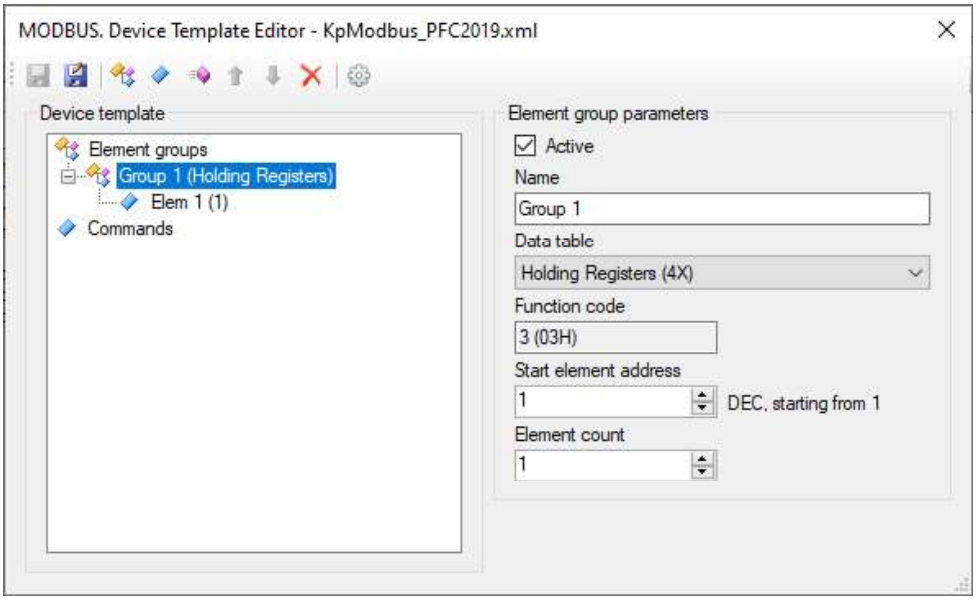



FIG. 5.5: Configuração de dispositivos utilizando o Rapid SCADA.

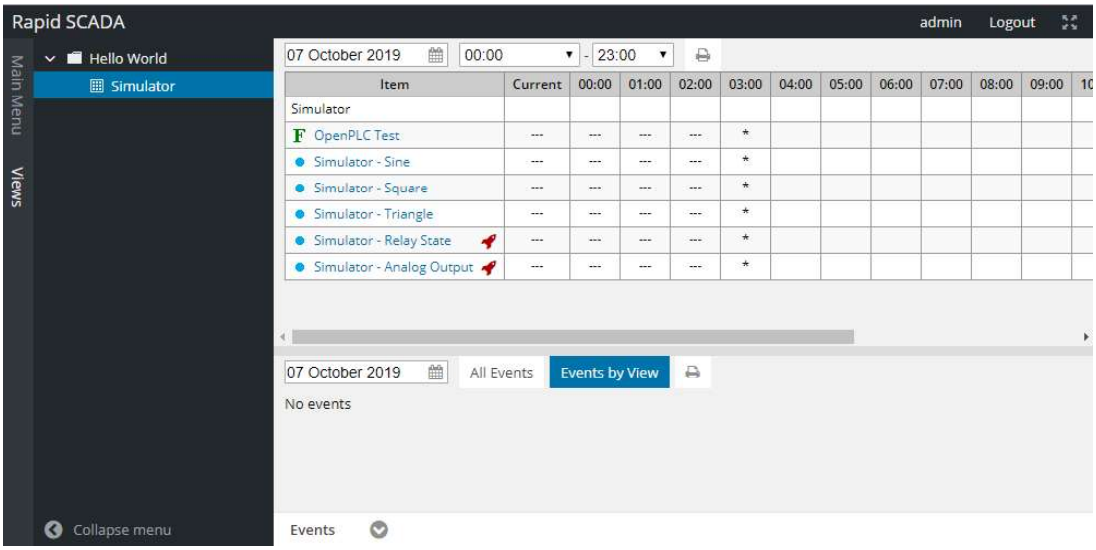



FIG. 5.6: Interface *web* do Rapid SCADA.

## 5.4 CAMADA DE CONTROLE

### 5.4.1 PLC

A configuração do PLC foi baseada na referência (ALVES, 2018). Esse programa emula uma PLC em uma máquina Linux. Para isso, foi realizada a instalação e configuração em uma máquina virtual com o sistema operacional Ubuntu 16.04, seguindo o padrão adotado no projeto. Dessa forma, foi criado um PLC virtual que utiliza o *software* OpenPLC para responder solicitações MODBUS/TCP provenientes do servidor SCADA.

A Figura 5.7 apresenta a página principal do emulador. Nesta é possível verificar o *status* da execução, interromper e retomar seu funcionamento. Além disso, possui interface para adicionar programas e arquivos de configuração específicos a serem executados.

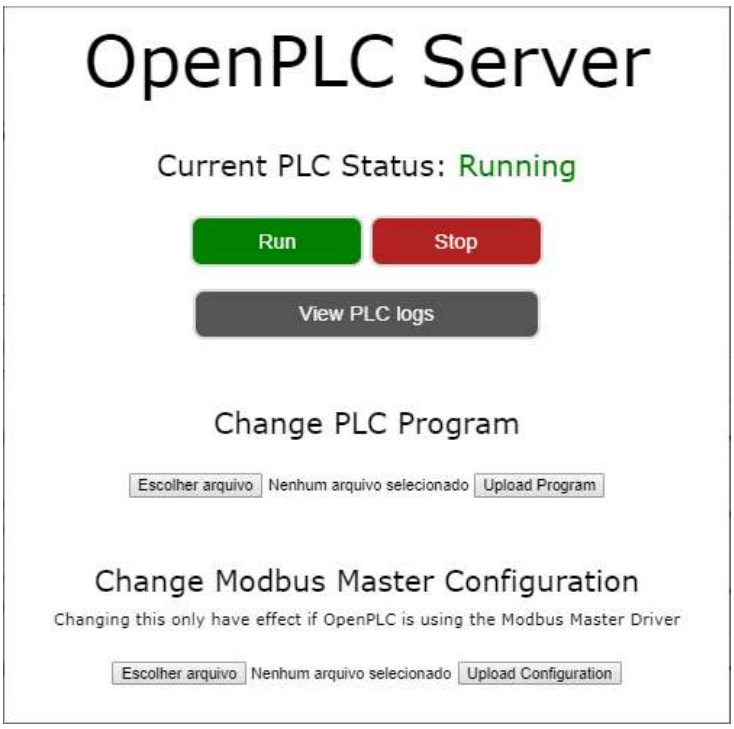



FIG. 5.7: Emulador OpenPLC.

## 5.5 GRUPOS DE SEGURANÇA

O OpenStack Neutron possui como funcionalidade a criação de grupos de segurança. Estes permitem realizar controle tanto de acesso às máquinas virtuais quanto de tráfego na rede, por meio da definição de políticas de segurança para faixas de endereços, portas e protocolos. Além disso, as regras de um grupo podem ser aplicadas sobre outro grupo. Por consequência, é possível que as políticas atuem diretamente sobre um conjunto dinâmico de máquinas que compõem outro grupo. Portanto, não é necessário atualizá-las nos eventos de alteração na arquitetura - criação ou exclusão de instâncias de máquinas.

Por padrão, a infraestrutura utiliza a política *deny all*, bloqueando o fluxo de todas as fontes. Dessa forma, as regras definidas pelo administrador do sistema são exceções que permitem tráfego de ingresso ou egresso em condições a serem definidas. Todas as instâncias criadas no OpenStack são associadas a um grupo de segurança *default*, com políticas de acesso irrestrito.

Neste projeto, este recurso é empregado como instâncias de *firewall* entre as regiões

e camadas da arquitetura desenvolvida. A Figura 5.8 apresenta a tela de configuração dos grupos de segurança acessada pelo Horizon. Seguindo a arquitetura de referência da Figura 2.1, foram definidas quatro conjuntos de regras além do grupo padrão.

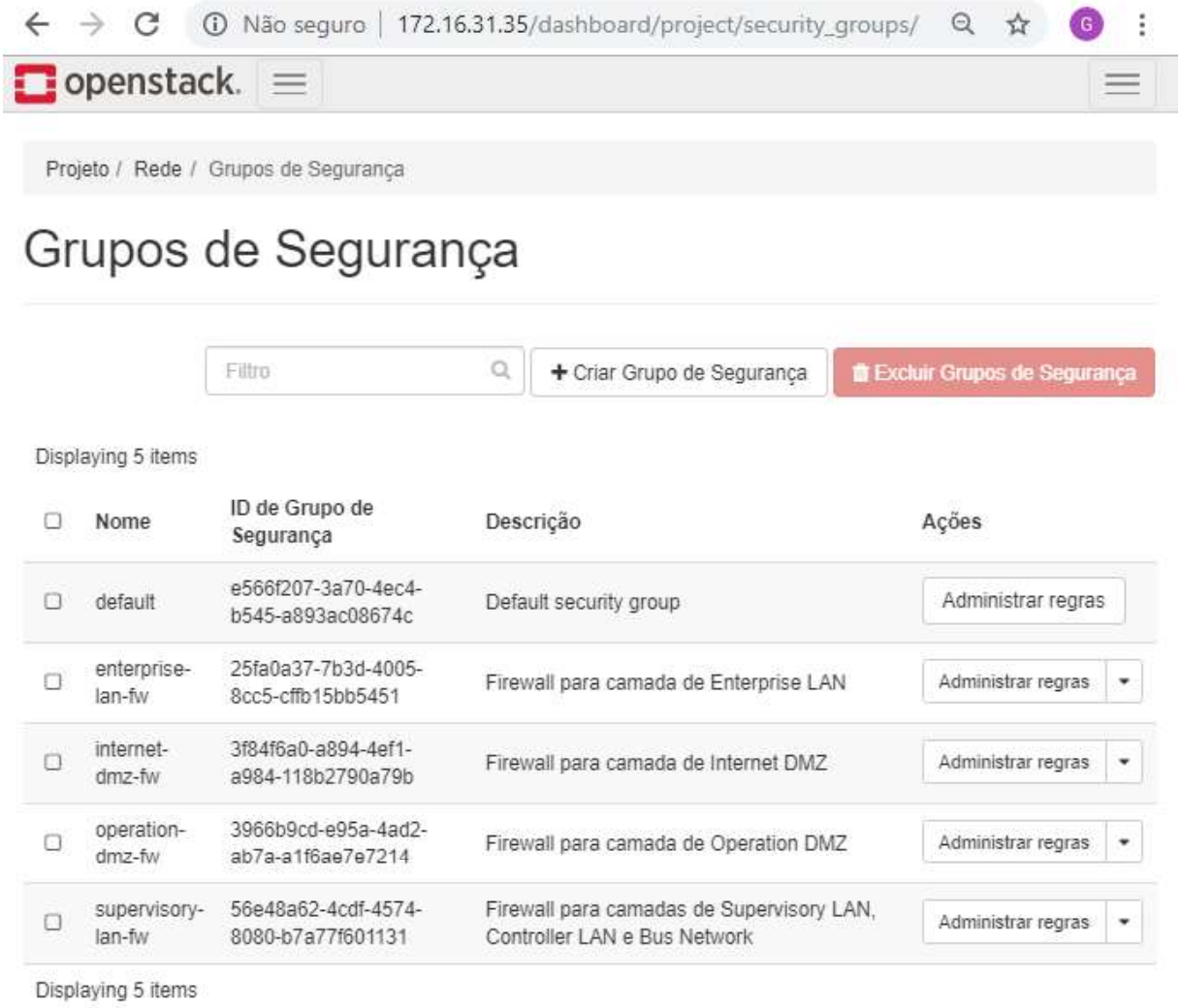



FIG. 5.8: *Firewall* implementado com grupos de segurança.

Para cada grupo de segurança listado anteriormente, são definidos conjuntos de regras. A Figura 5.9 apresenta a página de gerenciamento de regras do grupo de segurança *supervisory-lan-fw*, no qual foi permitido o ingresso e egresso dos protocolos TCP e UDP para tráfego com origem ou destino em máquinas que compõem o grupo de segurança *operation-dmz-fw*.

Os outros grupos estão definidos de maneira análoga. A Tabela 5.2 resume as políticas de acesso. A coluna de acesso representa as regiões e sub-redes com as quais cada grupo de segurança é permitido se comunicar.

Observa-se que as regras poderiam ser mais restritivas e específicas, possibilitando


FIG. 5.9: Regras do grupo de segurança *supervisory-lan-fw*.

TAB. 5.2: Políticas de *firewall* dos grupos de segurança.

| **Grupo de Segurança** | **Acesso** |
|---|---|
| internet-dmz-fw | internet pública, enterprise-lan |
| enterprise-lan-fw | internet-dmz, operation-dmz |
| operation-dmz-fw | enterprise-lan, supervisory-lan |
| supervisory-lan-fw | operation-dmz |

controle e segurança ainda maiores. Isso seria possível limitando os intervalos de portas e definindo grupos individuais para cada máquina.

Após a criação e configuração de todas os grupos e regras, passou-se para a etapa de validação. De forma a verificar as configurações realizadas, foram utilizadas máquinas de teste, uma em cada região e uma na sub-rede pública, com endereços IP definidos de forma arbitrária na criação das instâncias. Com isso, constatou-se o correto funcionamento dos grupos de segurança.

## 5.6 ARQUITETURA DE REDES

Utilizando os recursos do OpenStack Neutron, foram configuradas uma rede pública, para acesso à internet, e três redes privadas, interligados por um roteador. A rede *private* é dividida em duas sub-redes que definem as regiões *Internet-DMZ* e *Entrerprise LAN*. A rede *operation* engloba as três máquinas da região *Operation-DMZ*, enquanto a *sup_control*

conecta os componentes da *Supervisory LAN*.

Para este processo, foi necessário definir os endereços IP das redes, *gateway* e intervalos de alocação. A escolha destes seguiu um padrão, no qual foram alocados 256 endereços para cada sub-rede, de forma contígua. Além disso, o primeiro endereço é destinado ao *gateway*, e o intervalo de alocação vai de 2 à 254. Esses dados estão consolidados na Tabela 5.3.

No momento da criação de uma nova instância de máquina virtual, deve ser indicada a qual sub-rede ela pertence. Caso a rede esteja com a opção de DHCP habilitada, a escolha do endereço IP será feita de forma automática pelo OpenStack Neutron. Caso contrário, deverá também ser indicado o endereço manualmente. Em ambas situações, deve estar dentro do intervalo de alocação definido para a sub-rede em questão.

Com relação à rede pública, endereço foi definido a partir do endereço da *bridge* 'br-ex' (RED HAT, 2019) do servidor do OpenStack. Nesse projeto, o endereço é 172.24.4.1. A Figura 5.10 apresenta as configurações de rede expondo este endereço.


FIG. 5.10: Endereço da *bridge* 'br-ex'.

TAB. 5.3: Especificação dos endereços de rede.

| Rede | Sub-rede | Endereço | Gateway | Intervalo | Alocação |
|---|---|---|---|---|---|
| public | public-subnet | 172.24.4.0/24 | 172.24.4.1 | 172.24.4.0 - 172.24.4.255 | 172.24.4.2 - 172.24.4.254 |
| private | internet-dmz | 10.0.0.0/24 | 10.0.0.1 | 10.0.0.0 - 10.0.0.255 | 10.0.0.2 - 10.0.0.254 |
| private | enterprise-lan | 10.0.1.0/24 | 10.0.1.1 | 10.0.1.0 - 10.0.1.255 | 10.0.1.2 - 10.0.1.254 |
| operation | operation-dmz | 10.0.2.0/24 | 10.0.2.1 | 10.0.2.0 - 10.0.2.255 | 10.0.2.2 - 10.0.2.254 |
| sup_control | supervisory-lan | 10.0.3.0/24 | 10.0.3.1 | 10.0.3.0 - 10.0.3.255 | 10.0.3.2 - 10.0.3.254 |

Após a configuração, é possível analisar a rede obtida por meio do Horizon. São apresentadas três formas de visualização. A Figura 5.11 ilustra a rede do sistema por uma representação com grafo. As regiões sombreadas representam as redes privadas, as quais possuem interfaces com as máquinas virtuais criadas. A rede pública está conectada à internet e possibilita acesso externo.

Finalmente, as Figura 5.12 e Figura 5.13 fornecem outras representações da topologia de rede, nas quais é possível ver os endereços IP e identificação dos componentes. Na

primeira, é apresentada uma visualização consolidada, enquanto que a segunda é mais detalhada.

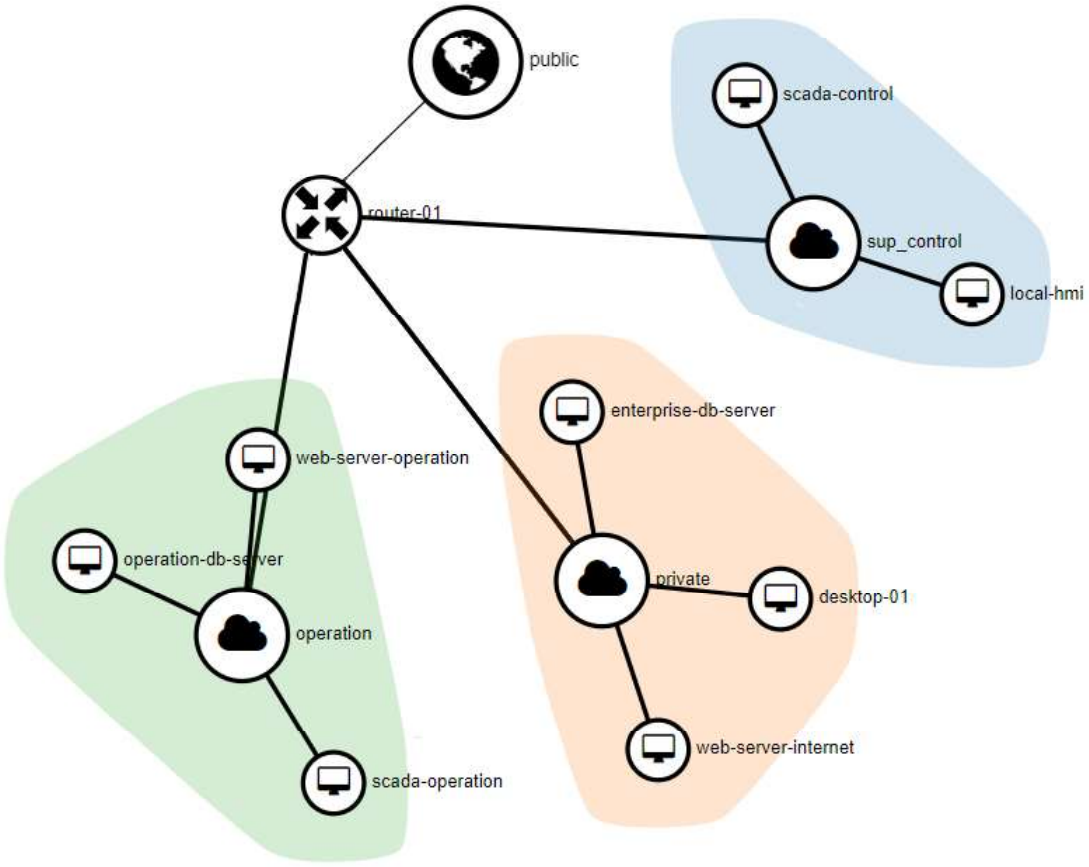



FIG. 5.11: Visualização em grafo da rede no Horizon.

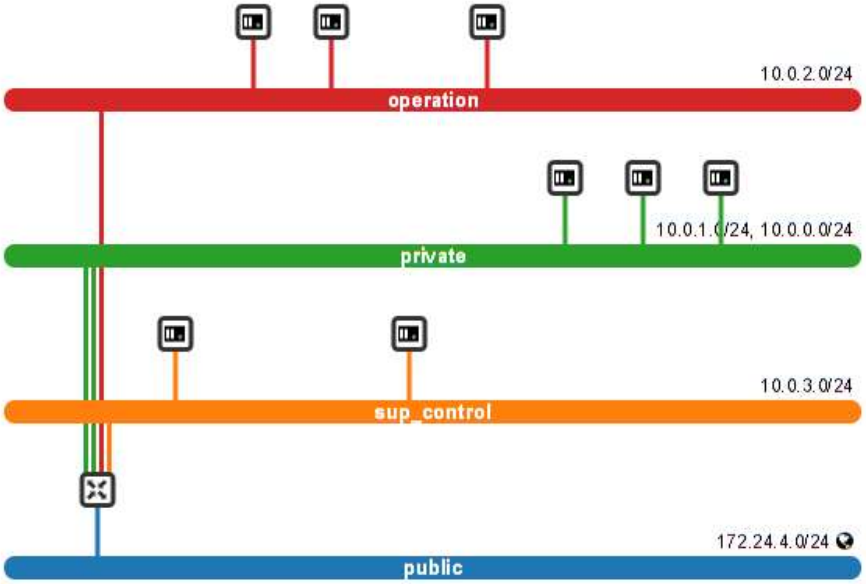



FIG. 5.12: Visualização em topologia consolidada da rede no Horizon.

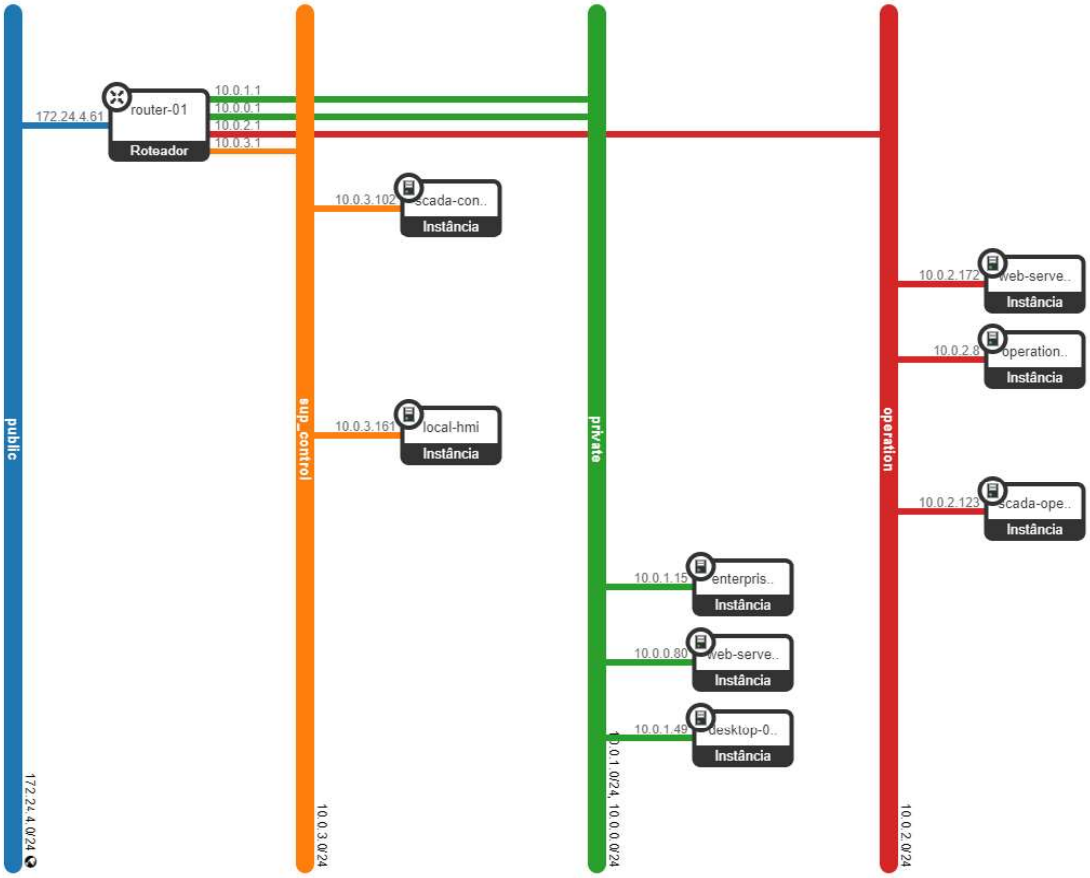



FIG. 5.13: Visualização em topologia completa da rede no Horizon.

# 6 TESTES E VALIDAÇÕES

Após a conclusão da estruturação da arquitetura, passando pela virtualização das máquinas de usuários, servidores, redes e políticas de segurança, foi iniciada a etapa de validação.

Este capítulo destina-se a apresentar os testes e validações realizados para assegurar o correto funcionamento dos componentes virtualizados no simulador de infraestrutura crítica. As seções a seguir detalham os procedimentos e expõem os resultados obtidos nas verificações.

## 6.1 SERVIDORES WEB E BANCO DE DADOS

O ambiente definido possui dois pares distintos de servidor *web* e servidor de banco de dados. Cada servidor *web* possui uma página PHP, cujo código é apresentado na Figura 6.1. Por meio deste, é realizada a conexão com o banco de dados remoto, atualização e consulta à tabela de eventos e exibição dos resultados. Dessa forma, para validar o funcionamento destes componentes, devem ser verificados os três seguintes itens.

```php
<?php
 $mysqli = new mysqli("10.0.1.35", "root", "senha1234", "logs");

 if ($mysqli->connect_errno){
  printf("Connect failed: %s\n", $mysqli->connect_error);
  exit();
 } else if (mysqli_connect_errno()) trigger_error(mysqli_connect_error());

 $sql = "insert into events (event_date) values (Now())";
 $query = $mysqli->query($qsl) or die(mysqli_error($mysqli));
?>
<html>
  <head>
    <title>Servidor web</title>
  </head>
  <body>
    <h1>PHP conectado com o servidor MySQL!</h1>

<?php
 $select = "select * from events";
 $queryC = $mysqli->query($select) or die(mysqli_error($mysqli));
 echo 'Registros encontrados: ' . $queryC->num_row . '<br>';

 while($row=mysqli_fetch_array($queryC)) {
   printf("%s - (%s)", $row[0], $row[1]);
   echo'<br>';
 }
?>

  </body>
</html>
```

FIG. 6.1: Página PHP para o servidor *web*.

- tanto o servidor *web* quanto o servidor de banco de dados podem ser acessados por outras máquinas, respeitando as regras dos grupos de segurança;
- quando houver uma requisição ao servidor *web*, este deve realizar uma inserção seguida de uma consulta na tabela de eventos do servidor de banco de dados; e
- o acesso ao servidor *web* deve apresentar ao usuário o resultado das consultas.

Para essa validação foi realizado o seguinte teste nos dois pares de servidores. O servidor *web* foi acessado pelo navegador, conforme a Figura 6.2. Com isso, atesta-se que a integração entre os componentes foi feita de maneira correta, pois foi estabelecida a conexão, houve atualização dos dados e posterior consulta, sendo os registros exibidos na tela. Realizando um novo acesso em seguida, verificou-se que foi incluído um novo registro na tabela. O servidor *web* pôde simultaneamente ser acessado por um cliente em outra máquina e acessar o servidor de banco de dados. Assim, são validados os três itens supracitados.

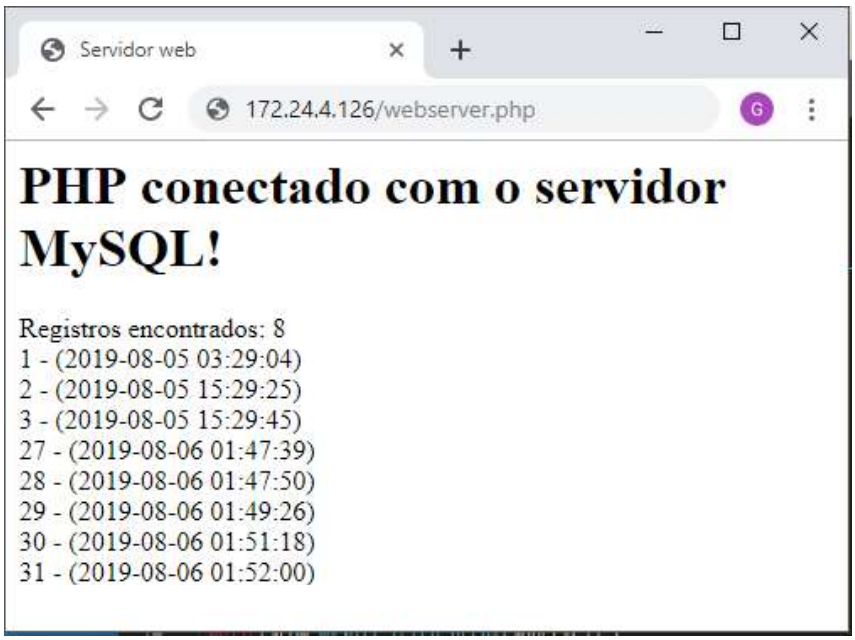



FIG. 6.2: Servidor *web* conectado ao servidor de banco de dados.

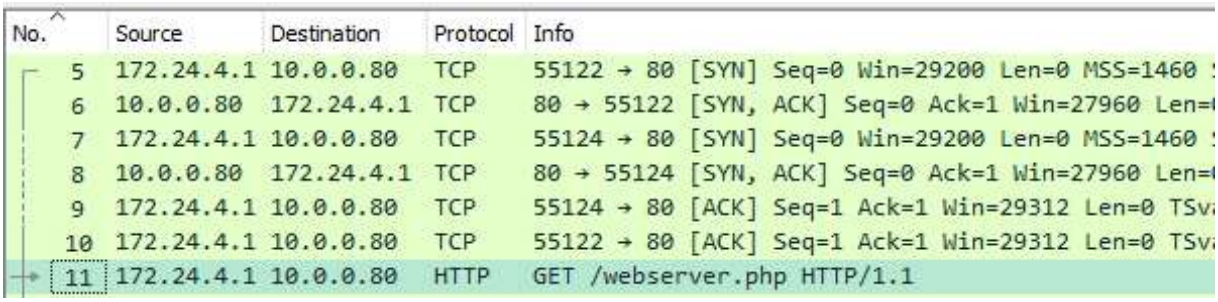
## 6.2 TRÁFEGO DE REDE

A validação da comunicação entre os componentes do sistema foi realizada utilizando o comando 'tcpdump' em conjunto com a ferramenta Wireshark[8]. Com isso, é possível

[8] `https://www.wireshark.org/`

capturar e analisar o tráfego de rede, verificando os protocolos empregados na conexão, comunicação e envio de pacotes. Nessa etapa, foram avaliados os diferentes serviços presentes no sistema: servidor *web*, banco de dados, SCADA e PLC.

Para o servidor *web*, foi realizado um acesso à página PHP, localizada no endereço '10.0.0.80'. A Figura 6.3 apresenta o tráfego de rede capturado nesse processo. Observa-se nas linhas 5, 6 e 10 os pacotes [SYN], [SYN, ACK] e [ACK] respectivamente, caracterizando o *handshake* de três vias do protocolo TCP. Após o estabelecimento da conexão, é realizada a requisição HTTP GET ao arquivo 'webserver.php', presente na linha 11.



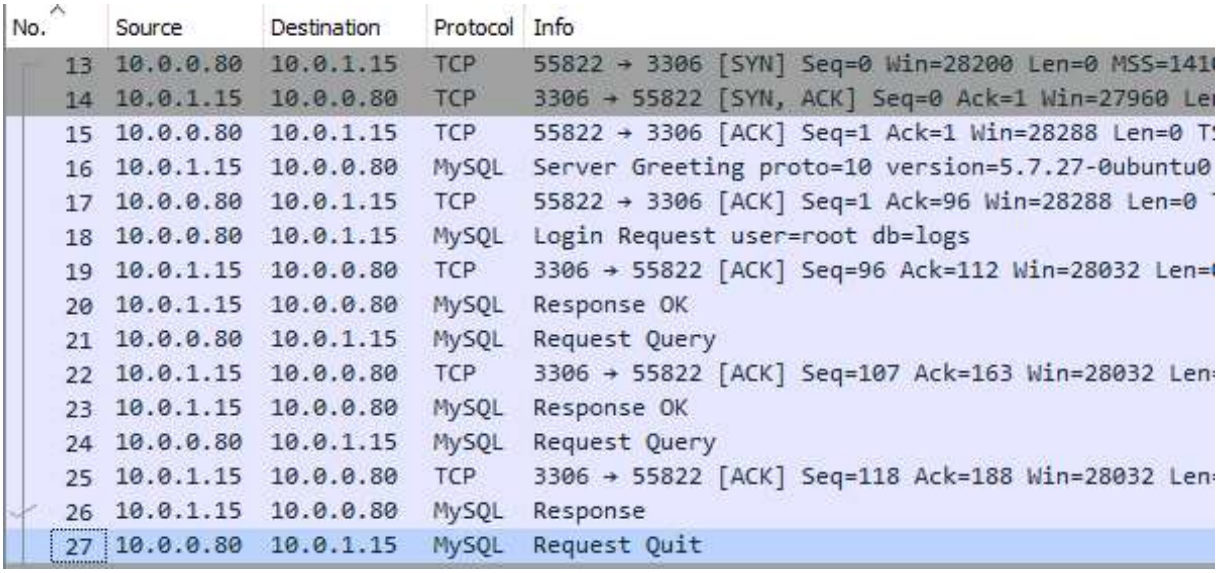
| No. | Source | Destination | Protocol | Info |
|---|---|---|---|---|
| 5 | 172.24.4.1 | 10.0.0.80 | TCP | 55122 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 |
| 6 | 10.0.0.80 | 172.24.4.1 | TCP | 80 → 55122 [SYN, ACK] Seq=0 Ack=1 Win=27960 Len= |
| 7 | 172.24.4.1 | 10.0.0.80 | TCP | 55124 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 |
| 8 | 10.0.0.80 | 172.24.4.1 | TCP | 80 → 55124 [SYN, ACK] Seq=0 Ack=1 Win=27960 Len= |
| 9 | 172.24.4.1 | 10.0.0.80 | TCP | 55124 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSv |
| 10 | 172.24.4.1 | 10.0.0.80 | TCP | 55122 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSv |
| 11 | 172.24.4.1 | 10.0.0.80 | HTTP | GET /webserver.php HTTP/1.1 |

FIG. 6.3: Tráfego de rede para o servidor *web*.

Com relação ao servidor de banco de dados, verifica-se, além da conexão TCP, o tráfego utilizando o protocolo MySQL, conforme ilustrado na figura 6.4. Este é dividido em três etapas. Primeiramente, é realizada a autenticação, na linha 18, com o envio das credenciais de acesso ao servidor, e a resposta '*Response OK*' na linha 20. Em seguida, são enviadas as *queries*, exibidas nas linhas 21 e 24, as quais são respondidas nas linhas 23 e 26, respectivamente. Por fim, a conexão é encerrada com o envio do pacote MySQL com '*Request Quit*', verificado na linha 27.

Para testar a integração do servidor SCADA e o PLC emulado, verifica-se o tráfego gerado utilizando o protocolo MODBUS/TCP. A captura deste é exposta na Figura 6.5. Além da conexão TCP com o *handshake* de três vias, que ocorre nas linhas 57, 58 e 60, a comunicação também é composta por pacotes que utilizam o protocolo MODBUS/TCP.

As linhas 61 e 69 da Figura 6.5 apresentam os pacotes enviados pelo servidor SCADA utilizando o protocolo MODBUS/TCP. Os destinos dos pacotes são compostos pelo endereço e porta da máquina que executa o emulador do PLC, conforme indicado anteriormente no arquivo de configuração exibido na Figura 5.4. Além disso, na descrição do pacote, verifica-se a utilização da função 03 (leitura de registros retentivos), de acordo com o que foi configurado e exposto na Figura 5.5. Devido ao emulador do PLC não ter sido configurado para executar um programa, não foi enviada nenhuma resposta específica. Entretanto, os pacotes de retorno [RST] (expostos nas linhas 62, 63 e 70), enviados pela



| No. | Source | Destination | Protocol | Info |
|---|---|---|---|---|
| 13 | 10.0.0.80 | 10.0.1.15 | TCP | 55822 → 3306 [SYN] Seq=0 Win=28200 Len=0 MSS=141 |
| 14 | 10.0.1.15 | 10.0.0.80 | TCP | 3306 → 55822 [SYN, ACK] Seq=0 Ack=1 Win=27960 Le |
| 15 | 10.0.0.80 | 10.0.1.15 | TCP | 55822 → 3306 [ACK] Seq=1 Ack=1 Win=28288 Len=0 T |
| 16 | 10.0.1.15 | 10.0.0.80 | MySQL | Server Greeting proto=10 version=5.7.27-0ubuntu0 |
| 17 | 10.0.0.80 | 10.0.1.15 | TCP | 55822 → 3306 [ACK] Seq=1 Ack=96 Win=28288 Len=0 |
| 18 | 10.0.0.80 | 10.0.1.15 | MySQL | Login Request user=root db=logs |
| 19 | 10.0.1.15 | 10.0.0.80 | TCP | 3306 → 55822 [ACK] Seq=96 Ack=112 Win=28032 Len= |
| 20 | 10.0.1.15 | 10.0.0.80 | MySQL | Response OK |
| 21 | 10.0.0.80 | 10.0.1.15 | MySQL | Request Query |
| 22 | 10.0.1.15 | 10.0.0.80 | TCP | 3306 → 55822 [ACK] Seq=107 Ack=163 Win=28032 Len |
| 23 | 10.0.1.15 | 10.0.0.80 | MySQL | Response OK |
| 24 | 10.0.0.80 | 10.0.1.15 | MySQL | Request Query |
| 25 | 10.0.1.15 | 10.0.0.80 | TCP | 3306 → 55822 [ACK] Seq=118 Ack=188 Win=28032 Len |
| 26 | 10.0.1.15 | 10.0.0.80 | MySQL | Response |
| 27 | 10.0.0.80 | 10.0.1.15 | MySQL | Request Quit |

FIG. 6.4: Tráfego de rede para o servidor MySQL.

máquina do emulador do PLC a partir da porta 502, indicam a integração entre os dois componentes.

| No. | Source | Destination | Protocol | Info |
|---|---|---|---|---|
| 57 | 10.0.3.219 | 10.0.3.18 | TCP | 57510 → 502 [SYN] Seq=0 Win=65535 Len=0 MSS=1360 SACK_PERM=1 |
| 58 | 10.0.3.18 | 10.0.3.219 | TCP | 502 → 57510 [SYN, ACK] Seq=0 Ack=1 Win=28200 Len=0 MSS=1410 SACK_PERM=1 |
| 59 | 10.0.3.219 | 10.0.3.18 | TCP | 57510 → 502 [RST] Seq=1 Win=0 Len=0 |
| 60 | 10.0.3.219 | 10.0.3.18 | TCP | 57510 → 502 [ACK] Seq=1 Ack=1 Win=65535 Len=0 |
| 61 | 10.0.3.219 | 10.0.3.18 | Modbus/TCP | Query: Trans: 1; Unit: 0, Func: 3: Read Holding Registers |
| 62 | 10.0.3.18 | 10.0.3.219 | TCP | 502 → 57510 [RST] Seq=1 Win=0 Len=0 |
| 63 | 10.0.3.18 | 10.0.3.219 | TCP | 502 → 57510 [RST] Seq=1 Win=0 Len=0 |
| 64 | 10.0.3.219 | 10.0.3.18 | TCP | 52484 → 502 [SYN] Seq=0 Win=65535 Len=0 MSS=1360 SACK_PERM=1 |
| 65 | 10.0.3.18 | 10.0.3.219 | TCP | 502 → 52484 [SYN, ACK] Seq=0 Ack=1 Win=28200 Len=0 MSS=1410 SACK_PERM=1 |
| 66 | 10.0.3.219 | 10.0.3.18 | TCP | 52484 → 502 [RST] Seq=1 Win=0 Len=0 |
| 67 | 10.0.3.219 | 10.0.3.18 | TCP | 52484 → 502 [ACK] Seq=1 Ack=1 Win=65535 Len=0 |
| 68 | 10.0.3.18 | 10.0.3.219 | TCP | 502 → 52484 [RST] Seq=1 Win=0 Len=0 |
| 69 | 10.0.3.219 | 10.0.3.18 | Modbus/TCP | Query: Trans: 1; Unit: 0, Func: 3: Read Holding Registers |
| 70 | 10.0.3.18 | 10.0.3.219 | TCP | 502 → 52484 [RST] Seq=1 Win=0 Len=0 |
| 71 | 10.0.3.219 | 10.0.3.18 | TCP | 43072 → 502 [SYN] Seq=0 Win=65535 Len=0 MSS=1360 SACK_PERM=1 |
| 72 | 10.0.3.18 | 10.0.3.219 | TCP | 502 → 43072 [SYN, ACK] Seq=0 Ack=1 Win=28200 Len=0 MSS=1410 SACK_PERM=1 |
| 73 | 10.0.3.219 | 10.0.3.18 | TCP | 43072 → 502 [RST] Seq=1 Win=0 Len=0 |
| 74 | 10.0.3.219 | 10.0.3.18 | TCP | 43072 → 502 [ACK] Seq=1 Ack=1 Win=65535 Len=0 |
| 75 | 10.0.3.18 | 10.0.3.219 | TCP | 502 → 43072 [RST] Seq=1 Win=0 Len=0 |

FIG. 6.5: Tráfego de rede com protocolo MODBUS/TCP.

# 7 CONCLUSÃO

Este projeto de fim de curso objetivou estruturar um ambiente computacional para simular infraestruturas críticas. Para isso, foi empregado o OpenStack para virtualização de recursos baseados na arquitetura SCADA (ENISA, 2016).

O trabalho proporcionou a exposição a novos conceitos, bem como o aprofundamento em outros tópicos já estudados no curso de graduação. Com base nos casos listados na motivação do projeto, percebe-se que infraestruturas críticas são fundamentais para o bom funcionamento da sociedade. Dessa forma, o estudo e desenvolvimento de ferramentas para manter sua operabilidade contínua são de suma importância.

Percebe-se que sistemas de supervisão e controle devem ser aplicados nesses ambientes a fim de assegurar a consistência dos serviços oferecidos. O emprego de padrões como o ISA-95 na arquitetura SCADA é essencial para garantir a robustez e segurança do sistema. Isso permite aos operadores do sistema maior supervisão e controle das operações.

Ao término da execução do projeto, verifica-se que o OpenStack oferece recursos de computação necessários adequados para a simulação de sistemas. Além disso, o OpenStack Neutron possui como funcionalidade a criação de grupos de segurança, que podem ser empregados como instâncias de *firewall*. Ademais, foi possível verificar como a utilização de processos de virtualização e computação em nuvem possibilita o desenvolvimento de sistemas escaláveis e de boa alocação dos recursos computacionais, com a preservação da integridade e segurança dos sistemas.

Como oportunidades de melhoria para trabalhos futuros, pode-se apontar a ampliação da arquitetura, com inclusão de mais máquinas específicas em cada região. Dessa forma, o sistema se aproxima da arquitetura de referência da Figura 2.1.

Também podem ser adicionadas outras soluções de segurança. Dois exemplos disso são a implementação de *firewall* e *proxy* com servidores dedicados a este fim e a instalação e configuração de *antimalwares* nas máquinas dos usuários. Dessa forma, é possível alcançar maior proteção ao sistema, o que é fundamental no contexto de infraestruturas críticas.

# 8 REFERÊNCIAS BIBLIOGRÁFICAS


ADEEB, P. A. **A Seminar Report on Security in Cloud Computing**. 2014. 25 f. Tese (Electronics and Communication Engineering) – National Institute of Technology, Calicute, Índia, 2014.

THIAGO ALVES. OpenPLC v2. Disponível em: <https://github.com/thiagoralves/OpenPLC_v2>. Acesso em: 01 out. de 2019.

BIST, M.; WARIYA, M. ; AGARWAL, A. Comparing delta, open stack and xen cloud platforms: A survey on open source iaas. In: IEEE INTERNATIONAL ADVANCE COMPUTING CONFERENCE (IACC), 3., 2013. **Proceedings**... [S.l.: s.n.], 2013, p. 96–100.

BOWEN, C.; BUENNEMEYER, T. ; THOMAS, R. Next generation scada security: best practices and client puzzles. **Annual IEEE SMC Information Assurance Workshop**, v. 6, 2005. Disponível em: <https://ieeexplore.ieee.org/document/1495984>. Acesso em: 20 mai. de 2019.

BROWN, G.; CARLYLE, M.; SALMERÓN, J. ; WOOD, K. Defending critical infrastructure. **INFORMS Journal on Applied Analytics**, v. 36, p. 530–544, 2006.

CHURCH, R. L.; SCAPARRA, M. P. ; MIDDLETON, R. S. Identifying critical infrastructure: The median and covering facility interdiction problems. **Annals of the Association of American Geographers**, v. 94, p. 491–502, 2004.

DANEELS, A.; SALTER, W. What is scada?. In: INTERNATIONAL CONFERENCE ON ACCELERATOR AND LARGE EXPERIMENTAL PHYSICS CONTROL SYSTEMS, 7., 1999. **Proceedings**... [S.l.: s.n.], 1999, p. 399–343.

PAULO SERGIO MELO DE CARVALHO. A DEFESA CIBERNÉTICA E AS INFRAESTRUTURAS CRÍTICAS NACIONAIS. Disponível em: <http://www.nee.cms.eb.mil.br/attachments/article/101/cibernetica.pdf>. Acesso em: 01 out. de 2019.

ECKHART, M.; EKELHART, A. Towards security-aware virtual environments for digital twins. In: 4TH ACM WORKSHOP, 4., 2018. **Proceedings**... [S.l.: s.n.], 2018, p. 61–72.

ENISA. **Communication network dependencies for ICS/SCADA Systems**. Heraklion: European Union Agency for Network and Information Security, 2016. 80 p. (Relatório Técnico).

FAROOQI, N. Testing a trust management system for cloud computing using simulation. **International Journal for Information Security Research**, v. 6, p. 636–542, 2016.

GORAN, V.; SIMJANOSKA, M.; RISTOV, S. ; GUSEV, M. Business case: From iaas to saas. In: SMES DEVELOPMENT AND INNOVATION: BUILDING COMPETITIVE FUTURE OF SEE, 1., 2014. **Proceedings**... |S.l.: s.n.|, 2014, p. 801–808.

INTERNATIONAL SOCIETY OF AUTOMATION. Standard ISA-95. Disponível em: <https://isa-95.com/>. Acesso em: 20 mai. de 2019.

INVASÃO. Vírus Stuxnet danificou 10% das centrífugas de urânio do Irã. Disponível em: <http://www.invasao.com.br/2010/12/29/virus-stuxnet-danificou-10-das-centrifugas-de-uranio-do-ira/>. Acesso em: 18 set. de 2019.

J. POPEK, G.; P. GOLDBERG, R. Formal requirements for virtualizable third generation architectures. **Commun. ACM**, v. 17, p. 412–421, 1974.

MAGLARAS, L. A.; KIM, K.-H.; JANICKE, H.; FERRAG, M. A.; RALLIS, S.; FRAGKOU, P.; MAGLARAS, A. ; J.CRUZ, T. Cyber security of critical infrastructures. **ICT Express**, v. 4, n. 1, p. 42–45, 2018.

MERABTI, M.; KENNEDY, M. ; HURST, W. Critical infrastructure protection: A 21st century challenge. **International Conference on Communications and Information Technology (ICCIT)**, v. 13, 2011. Disponível em: <https://ieeexplore.ieee.org/document/5762681>. Acesso em: 20 jun. de 2019.

MINISTÉRIO DA DEFESA. DOUTRINA MILITAR DE DEFESA CIBERNÉTICA. Disponível em: <https://www.defesa.gov.br/arquivos/legislacao/emcfa/publicacoes/doutrina/md31_m_07_defes Acesso em: 01 out. de 2019.

OPENSTACK. Build the future of Open Infrastructure. Disponível em: <https://www.openstack.org>. Acesso em: 20 abr. de 2019.

RAPID SCADA. Rapid SCADA Documentation Version 5.7. Disponível em: <http://doc.rapidscada.net/content/latest/en/>. Acesso em: 25 ago. de 2019.

RED HAT. CHAPTER 9. CONNECT AN INSTANCE TO THE PHYSICAL NETWORK. Disponível em: <https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/10/html/networking_guide/sec-connect-instance>. Acesso em: 02 set. de 2019.

RINALDI, S. M.; PEERENBOOM, J. P. ; KELLY, T. K. Identifying, understanding, and analyzing critical infrastructure interdependencies. **IEEE Control Systems Magazine**, v. 21, n. 6, p. 11–25, 2001.

RAQUEL SODRÉ. 8 apagões que ficaram na história. Disponível em: <https://super.abril.com.br/blog/superlistas/8-apagoes-que-ficaram-na-historia/>. Acesso em: 06 set. de 2019.

TANENBAUM, A. S.; STEEN, M. V. Processes. In: _____. **Distributed Systems Principles and Paradigms**. Upper Saddle River: Pearson Prentice Hall, 2006. p. 69–114.

TECMUNDO. Malware causa blecaute em centenas de milhares de casas na Ucrânia. Disponível em: <https://www.tecmundo.com.br/ataque-hacker/92401-malware-causa-blecaute-centenas-milhares-casas-ucrania.htm>. Acesso em: 24 set. de 2019.

UHLEMANN, T. H. J.; LEHMANN, C. ; STEINHILPER, R. The digital twin: Realizing the cyber-physical production system for industry 4.0. In: THE 24TH CIRP CONFERENCE ON LIFE CYCLE ENGINEERING, 24., 2017. **Proceedings**... [S.l.]: Elsevier, 2017, p. 335–340.

WEN, X.; GU, G.; LI, Q.; GAO, Y. ; ZHANG, X. Comparison of open-source cloud management platforms: Openstack and opennebula. In: INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS AND KNOWLEDGE DISCOVERY, 9., 2012, Sichuan. **Electronic proceedings**... Piscataway: IEEE, 2012, p. 29–31. Disponível em: <http://ieeexplore.ieee.org/document/6234218>. Acesso em: 22 abr. de 2019.

YOUNIS, Y. A.; MERABTI, M. ; KIFAYAT, K. Secure cloud computing for critical. infrastructure: A survey.. In: ANNUAL POSTGRADUATE SYMPOSIUM ON THE

CONVERGENCE OF TELECOMMUNICATIONS, NETWORKING AND BROADCASTING, 14., 2013. **Proceedings**... [S.l.: s.n.], 2013, p. 1–6.

YUSTA, J. M.; CORREA, G. J. ; LACAL-ARÁNTEGUI, R. Methodologies and applications for critical infrastructure protection: State-of-the-art. **Energy Policy**, v. 39, p. 6100–6119, 2011.

# 9 APÊNDICES

# APÊNDICE 1: INSTALAÇÃO DO OPENSTACK

Para iniciar o desenvolvimento do projeto, foi realizada a instalação do OpenStack 16.0.0 baseada nas documentações *OpenStack Installation Guide*[9] e *DevStack*[10]. As instalações realizadas, configurações e alterações necessárias estão descritos nos passos a seguir. Também são documentadas dificuldades técnicas enfrentadas ao longo do processo e as soluções encontradas para contorná-las.

Passo 1: Definição do ambiente

O processo descrito a seguir realiza alterações consideráveis no sistema, o que pode comprometer funcionalidades e impedir a execução de determinadas aplicações. Dessa forma, recomenda-se que seja apenas realizada em servidores ou máquinas virtuais destinadas exclusivamente a este fim.

A instalação foi realizada em máquinas com 8 GB de memória e Ubuntu 16.04. A escolha do sistema operacional se deve ao fato de este ser o mais testado e possuir maior suporte *online* disponível para o OpenStack.

Passo 2: Adicionar usuário *stack*

Apesar de não ser um passo listado como obrigatório, não foi obtido êxito na primeira tentativa de instalação, na qual o usuário não foi criado. Dessa forma, recomenda-se que essa etapa seja executada.

Para criar um usuário, os seguintes comandos devem ser executados.

```
$ sudo useradd -s /bin/bash -d /opt/stack -m stack
$ echo ''stack ALL = (ALL) NOPASSWD: ALL'' | sudo tee
```

A resposta do sistema para estes deve ser 'stack ALL=(ALL) NOPASSWD: ALL', indicando a criação do usuário *stack* com os acessos necessários. Caso a mensagem seja diferente, deve-se verificar as permissões do usuário ativo. Essa etapa é finalizada pelo acesso como super-usuário.

```
$ sudo su - stack
```

[10] https://docs.openstack.org/install-guide/

[10] https://docs.openstack.org/devstack/latest/

Passo 3: *Download* do OpenStack

Os arquivos de instalação e configuração serão obtidos utilizando git por meio do repositório Devstack.

```
$ git clone https://opendev.org/openstack/devstack
$ cd devstack
```

Caso o usuário não possua git instalado, este pode ser obtido a partir do comando.

```
$ sudo apt-get install git
```

Passo 4: Configuração pré-instalação

Faz-se necessário a criação de um arquivo de configuração *local.conf* no diretório raiz do *devstack*. Neste devem estar listadas as configurações necessárias para a instalação. Apesar de os guias utilizados como referência apresentarem uma configuração mínima, foi necessário complementar o arquivo com outras instruções referentes a *logging* e ao Swift para contornar falhas na próxima etapa. A Figura 9.1 apresenta o arquivo de configuração utilizado para a instalação.


FIG. 9.1: Arquivo de configuração mínima para instalação.

Passo 5: Instalação

A instalação ocorre a partir do diretório raiz do *devstack* com a execução do comando a seguir e possui duração aproximada de 30 minutos (podendo variar de acordo com a conexão com a internet).

```
$ ./stack.sh
```

Alguns erros ocorreram durante essa etapa. Estes, bem como as ações realizadas para resolvê-los, estão resumidos a seguir.

- Erro de versão do *pip*: no arquivo *cap-pip.txt* há uma exigência de versão do *pip*, que é atualizada durante esse passo. Isso impedia o prosseguimento do processo devido à incompatibilidade de versões: foi regredida de 19.1.1 para 9.0.1, apesar de ser a mais recente no momento. Com isso, a instalação foi interrompida. A solução para este problema foi alterar a linha '*pip!=8, <10*' para '*pip > 19*' no arquivo.
- Falha durante a instalação do OpenStack Glance ('*g-api did not start*'): a instalação deste componente foi realizada de forma separada utilizando o tutorial completo do OpenStack. Após isso, a instalação padrão foi retomada utilizando *devstack*.

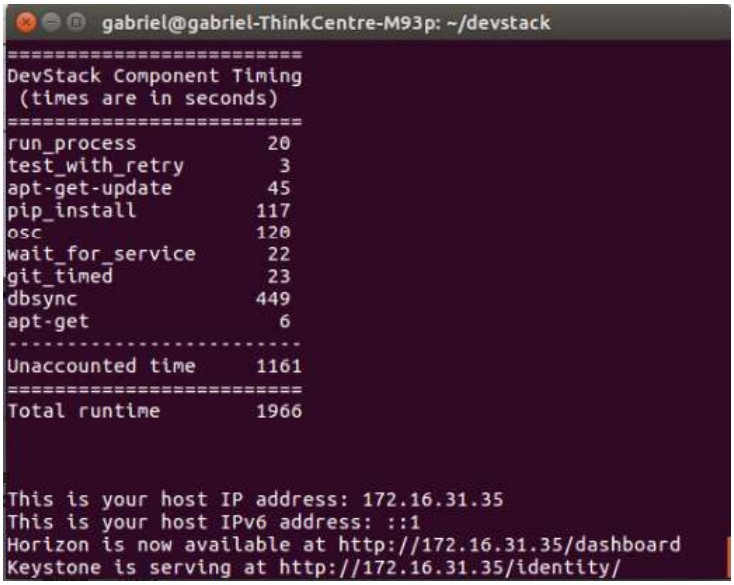



FIG. 9.2: Conclusão da instalação do OpenStack.

Ao término da instalação, são exibidas no console informações do processo de instalação e endereços IP para os componentes *dashboard* e provedor de identidade, conforme a Figura 9.2.

Por fim, recomenda-se ainda a execução dos seguintes comandos.

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get dist-upgrade
$ sudo apt-get autoremove
$ sudo apt-get autoclean
```

Passo 6: Conclusão e acesso

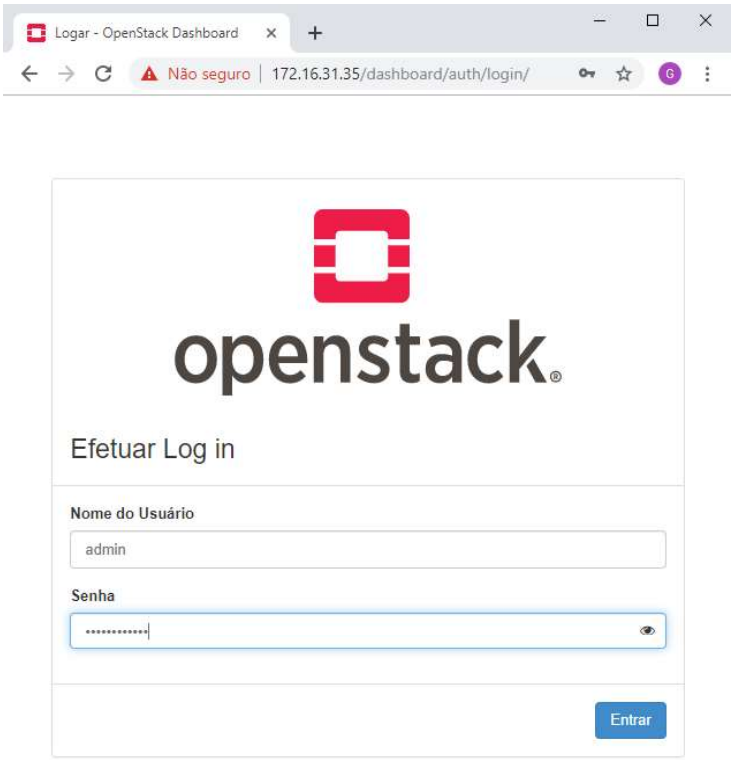



FIG. 9.3: Tela de login do *dashboard* do OpenStack.

O OpenStack Horizon pode ser acessado pela página exibida na Figura 9.3, utilizando o usuário 'admin' e a senha salva no arquivo de configuração *local.conf* (no caso dessa instalação, 'openstack123').

As informações de cada um dos componentes configurados e instalados é visível na aba correspondente, conforme exposto na Figura 9.4. A partir disso, pode-se validar a instalação por meio do acesso a cada componente por meio dos *endpoints* listados.

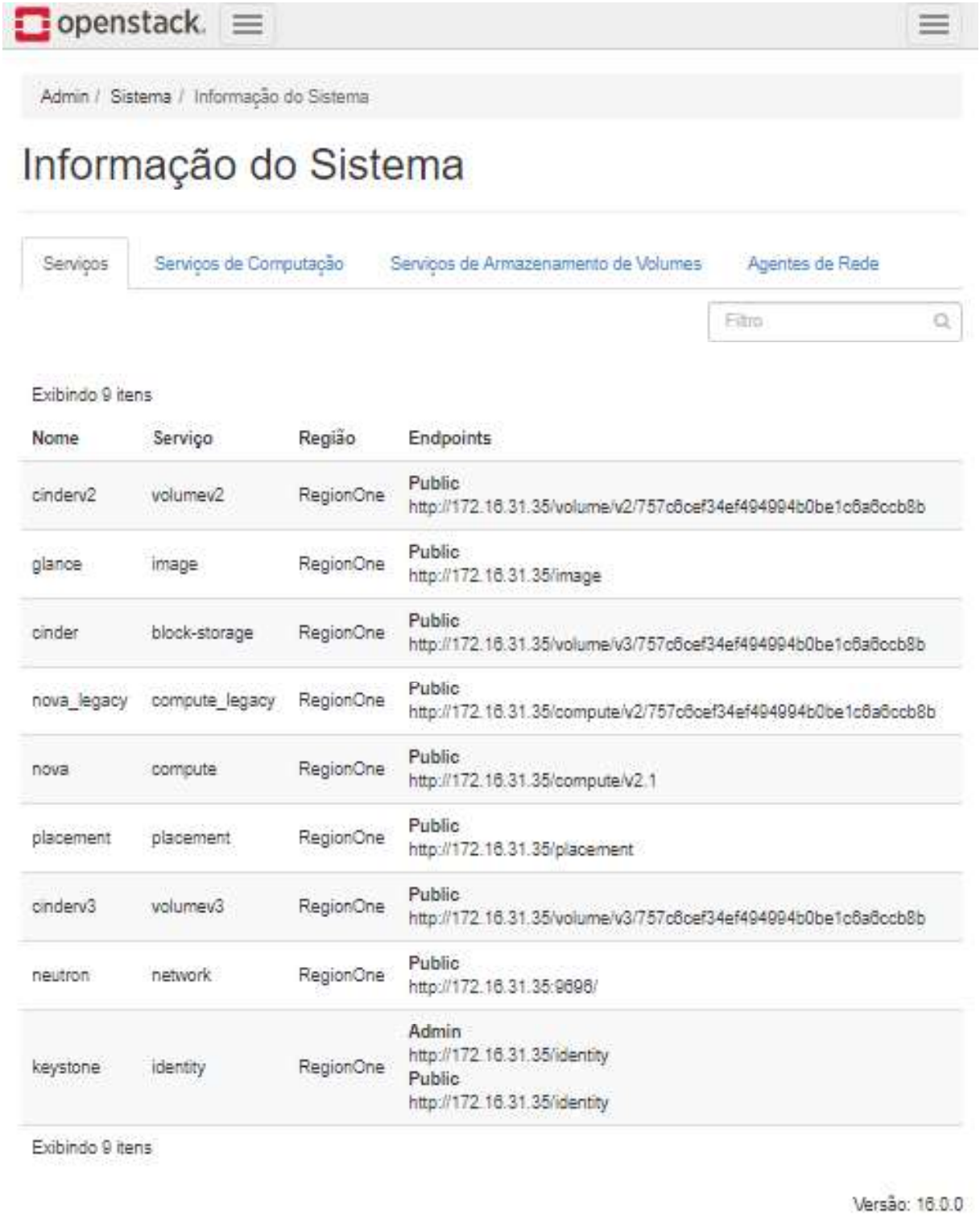



FIG. 9.4: Informação do sistema do OpenStack.

# APÊNDICE 2: CRIAÇÃO DE MÁQUINAS VIRTUAIS

O processo de criação de máquinas virtuais no painel de controle Horizon do OpenStack envolve a execução de até três etapas. A seguir estão listadas as ações adotadas e padronizadas para esse processo durante a execução do projeto. O exemplo ilusta o processo para o Ubuntu 16.04.6, entretanto é válido para os demais sistemas operacionais e imagens.

Passo 1: Imagens

Inicialmente, é necessário realizar o *download* das imagens dos sistemas operacionais. Para o projeto, foi utilizado como referência o guia de imagens da documentação do OpenStack [11], no qual está descrito como obter as imagens dos principais sistemas.

Em seguida, é necessário importar o arquivo no OpenStack pela tela de gerenciamento de imagens, etapa ilustrada na figura 9.5.

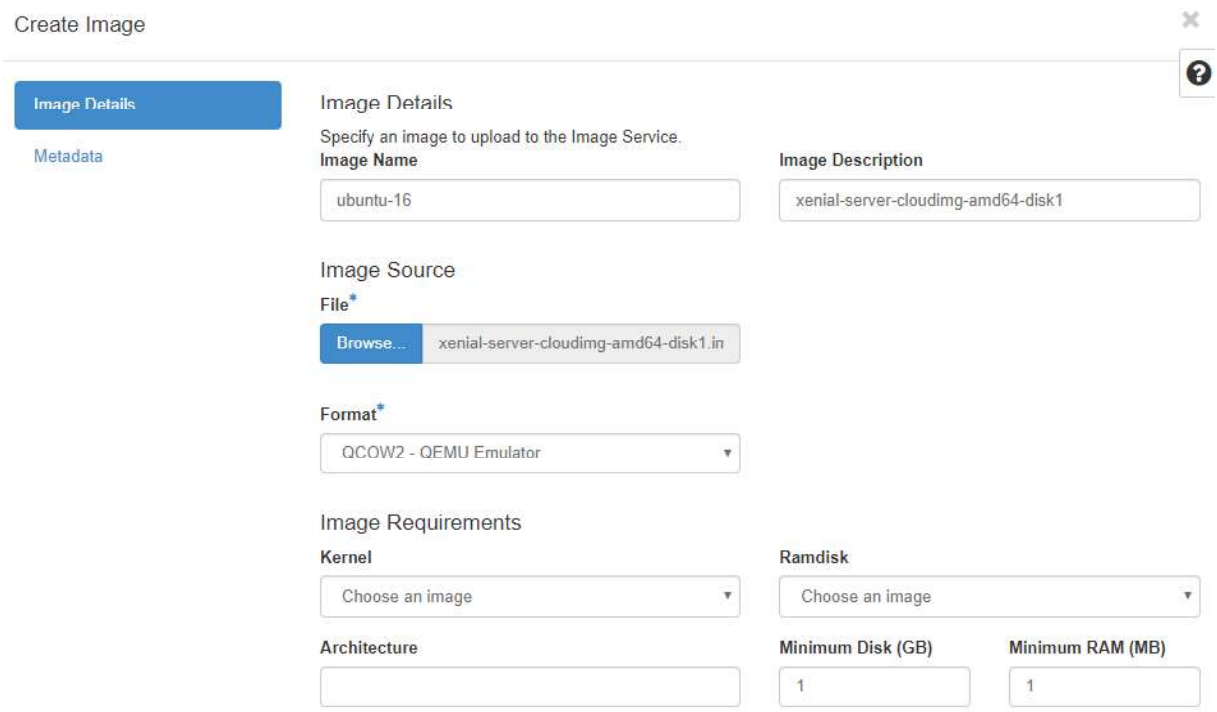



FIG. 9.5: Criação de imagem no OpenStack.

Passo 2: Instâncias

[11] `https://docs.openstack.org/image-guide/obtain-images.html`

Após a criação da imagem, parte-se para o processo de criação das instâncias das máquinas virtuais. Na página de gerenciamento de instâncias, clicar no botão de disparar instância. Após isso, devem ser preenchidos os dados gerais da máquina, passo ilustrado na Figura 9.6. Na aba 'Origem', deve ser alocada uma imagem, no caso a que foi adicionada ao projeto na etapa anterior. Na Figura 9.7, é representada a alocação da imgem do Ubuntu. Na aba de 'Flavor', deve ser selecionado um padrão de acordo com as necessidades da máquina. Os *flavors* são padrões pré-definidos de memória e capacidade de armazenamento da instância. Por fim, deve ser selecionada a rede à qual a instância está conectada. Após isso, a máquina pode ser acessada pela página de instâncias no Horizon, como ilustrado na Figura 9.8.

Para acesso às máquinas Ubuntu, é necessário definição de senha ou uso de par de chaves. Nesse projeto, no momento de criação das máquinas, foi adicionado o seguinte *script* na aba de 'Configuração', para que a senha fosse alterada no primeiro acesso.

```
#cloud-config
password: ubuntu
chpassword: { expire: False }
ssh_pwauth: True
```

Passo 3: Configuração de Rede

Nesse projeto foram utilizadas máquinas com os sistemas operacionais Ubuntu e Windows. Para as primeiras, não foi necessário realizar nenhuma configuração adicional de redes nas instâncias. Nesse caso, a atribuição de endereço IP e conexão com a rede foi feita de forma automática.

Entretanto, no caso das máquinas Windows, foi necessário alterar as configurações de rede e internet. Isso foi realizado no menu de propriedades de *ethernet* e IP, localizado em 'Control Panel > Network and Internet > Network and Sharing Center > Ethernet Status > Ethernet Properties'. A Figura 9.9 apresenta essa tela para uma instância criada. Foi necessário preencher os endereços da máquina, do *gateway* e dos servidores de DNS, conforme as configurações de rede realizadas no OpenStack.

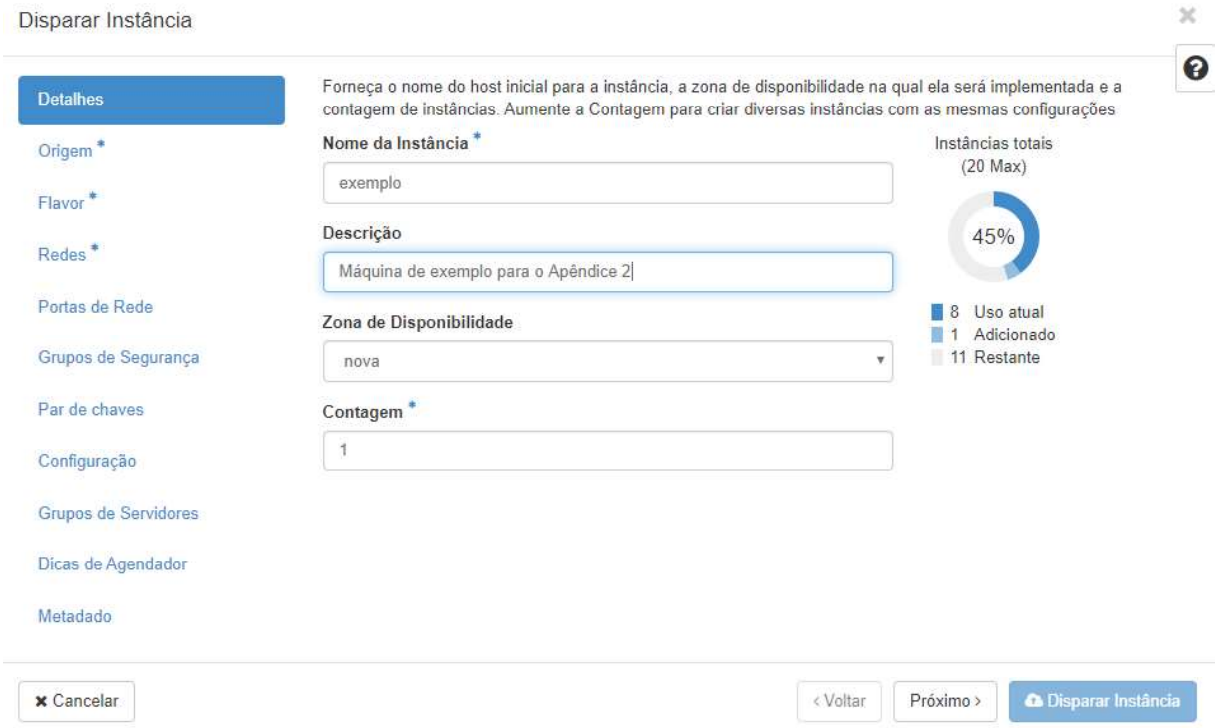



FIG. 9.6: Criação da instância no Horizon.


FIG. 9.7: Seleção de imagem para criação de instância no Horizon.

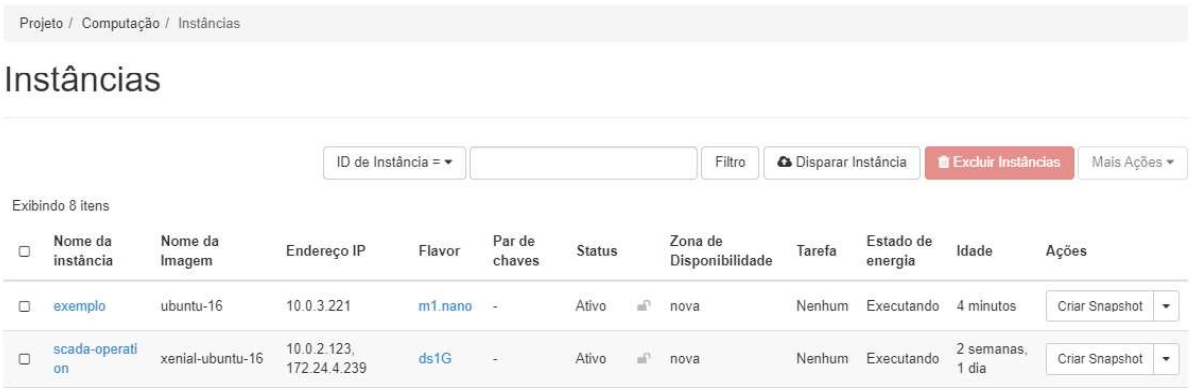



FIG. 9.8: Máquina virtual criada na página de instâncias no Horizon.

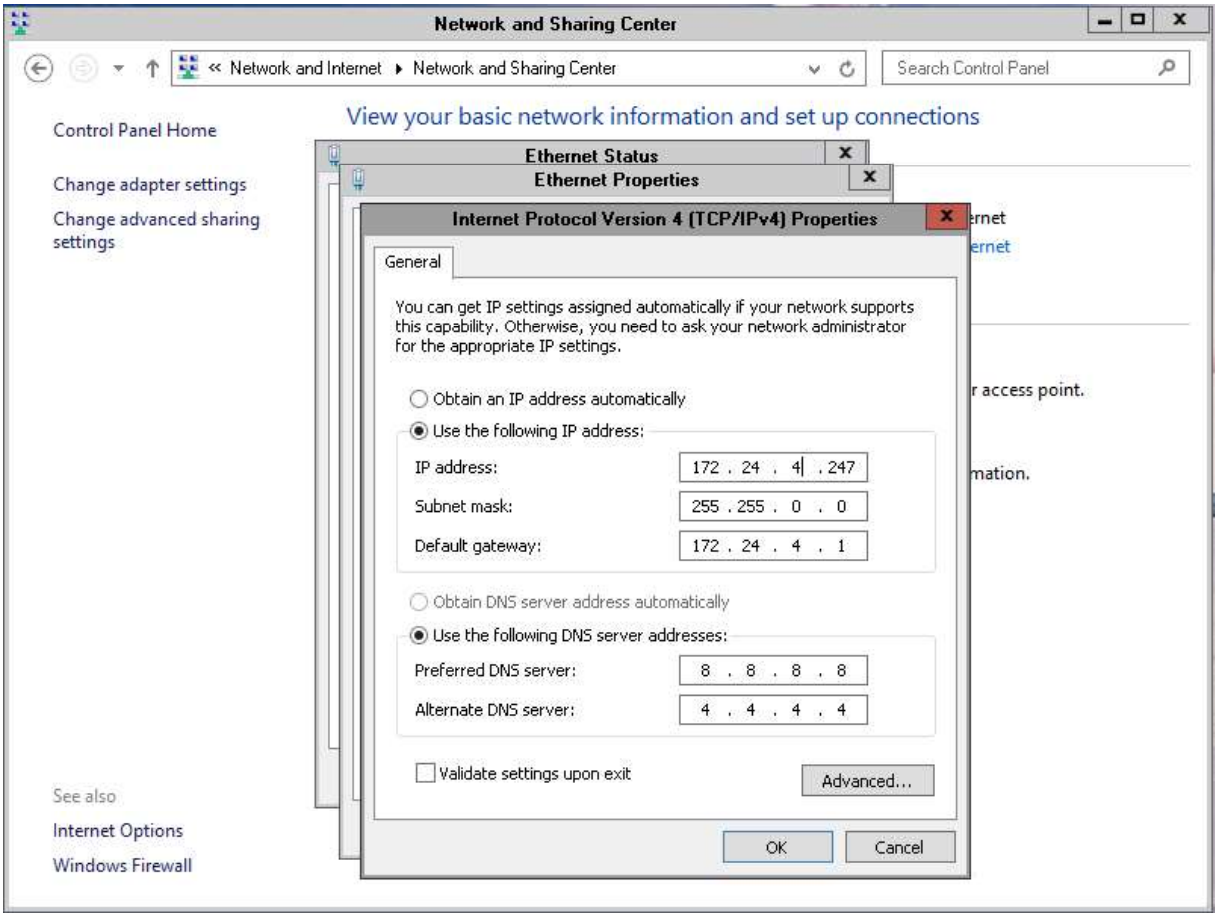



FIG. 9.9: Configuração de rede em máquina virtual Windows.