



UNIVERSIDADE FEDERAL DE MINAS GERAIS
CURSO DE ENGENHARIA DE SISTEMAS

ANÁLISE DE DADOS UTILIZANDO *cluster* DE BAIXO CUSTO: TENDÊNCIAS DE CONSUMO DA AZITROMICINA NO BRASIL ANTES E DURANTE A PANDEMIA DA COVID-19

FELIPE FONSECA ROCHA

Orientador: Ítalo Fernando Scotá Cunha
Universidade Federal de Minas Gerais

BELO HORIZONTE
JULHO DE 2022

FELIPE FONSECA ROCHA

**ANÁLISE DE DADOS UTILIZANDO *cluster* DE BAIXO
CUSTO: TENDÊNCIAS DE CONSUMO DA AZITROMICINA NO
BRASIL ANTES E DURANTE A PANDEMIA DA COVID-19**

Trabalho de Conclusão de Curso II apresentado ao Curso de graduação em Engenharia de Sistemas do Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do título de Bacharel em Engenharia de Sistemas.

Orientador: Ítalo Fernando Scotá Cunha
Universidade Federal de Minas Gerais

UNIVERSIDADE FEDERAL DE MINAS GERAIS
CURSO DE ENGENHARIA DE SISTEMAS
BELO HORIZONTE
JULHO DE 2022

Dedico este trabalho àquela que esteve comigo na jornada dessa graduação, minha força de vontade, minha motivação, minha esposa. Dedico também aos meus avós, pelo suporte aos meus estudos e modelos de cidadãos e chefes de família, mas que não puderam ver a conclusão. E também aos meus pais que me incentivaram a nunca desistir.

Agradecimentos

Agradeço a minha esposa, pelo amor e carinho, seu acolhimento nos dias difíceis, pelos lanches de incentivo, por nunca me deixar desistir, por sempre me acalmar quando o mundo girava mais rápido e mais feio do que eu dava conta, que em seu apoio incondicional me fez chegar a um ponto final nesse capítulo: graduação.

Ao meu orientador, Prof. Dr. Ítalo Fernando Scotá Cunha pela ajuda durante o processo de formulação de um trabalho válido e suas contribuições a este.

Aos docentes exemplares do curso de Engenharia de Sistemas que em muito contribuíram para minha formação.

À Prof. Dra. Ana Liddy Cenni de Castro Magalhães, por sua excelente coordenação e ao secretário do Colegiado de Engenharia de Sistemas, Júlio César Pereira de Carvalho, por sua disponibilidade e suporte.

Ao Centro de Estudos de Medicamentos da Faculdade de Farmácia da UFMG e todos os seus membros, por sua iluminação sobre o que é a extensão aliada ao ensino e orientada à pesquisa.

Aos colegas do curso que dividiram seus conhecimento e experiências.

À minha família, pela força motriz de tudo o que construo.

Aos meus amigos, um grupo seleto de nerds e suficientemente pacientes. Dentre eles graduados, mestres e doutores pela sua importância ao longo da minha trajetória.

"Os livro têm muita coisa escrita, é o que ele diz

Em pleno berço de Machado de Assis, olha o que eu sou obrigado a ouvir

Nega a ciência, esconde a doença, só negligência

Essa é a sentença pra falta de consciência que colocou ele ali"

(Chico César, DJ Caique, Rashid, Diário de Bordo 6)

Resumo

A análise de grandes volumes de dados, por vezes, requer um ou mais computadores de alto desempenho tornando viável a extração de informações. Na área da pesquisa, principalmente em instituições públicas, há uma restrição orçamentária que, nos últimos anos, apresentou uma diminuição crescente dos recursos disponibilizados, inviabilizando a aquisição de computadores que possibilitam tais análises. Uma alternativa à necessidade de computadores de alto desempenho é o uso de técnicas de computação distribuída. Assim, foi realizada a orquestração de recursos em *cluster* Kubernetes® para o processamento e a análise de dados. Além disso, avaliou-se a viabilidade dessa solução sob aspectos de desempenho, complexidade e viabilidade econômica, tendo como carga de trabalho a análise dos dados de consumo de azitromicina, no Brasil, entre os anos de 2014 e 2020. Oito computadores foram utilizados na estruturação do cluster, somando 42 núcleos e 88 GiB RAM. Para facilidade de configuração, mantendo a segurança dos clusters, todos foram configurados com chaves SSH para acesso remoto. A utilização de *desktops* na composição do *cluster* - que continuará disponível para uso no Departamento de Ciências da Computação - viabilizou o processamento de mais de 70 GB de dados de maneira distribuída, em 53 minutos, sem nenhum custo adicional com a aquisição de equipamentos para análise de dados. Quanto aos dados de consumo da azitromicina, no Brasil, observou-se um aumento das taxas de prescrição por 1.000 habitantes ao longo da série temporal de 66,2 para 83,8 prescrições, bem como em quase todas as unidades federativas, com destaque para Minas Gerais (66,2 para 149,2 prescrições por 1.000 habitantes), Rondônia (37,5 para 109,4 prescrições por 1.000 habitantes) e Roraima (37,2 para 99,8 prescrições por 1.000 habitantes). Comportamento semelhante também foi observado, na análise comparativa entre os anos pré e durante a pandemia (2019 e 2020), com aumento de 59,9 para 83,8 prescrições por mil habitantes, no país. A realização desse trabalho demonstrou que a utilização de *cluster* Kubernetes® é uma alternativa promissora para análise de grandes volumes de dados, especialmente sob o ponto de vista de utilização de recursos subutilizados, com o processo de recrutamento através de configuração dinâmica de gerenciadores de configurações. E que, considerando os dados analisados, aparentemente a pandemia da COVID-19 - incluindo a divulgação da azitromicina nos kits COVID - pode ter influenciado no maior aumento do consumo desse medicamento.

Palavras-chave: Kubernetes®. Virtualização. Contêineres. Hypervisor Tipo 2. Análise de dados.

Abstract

The analysis of large volumes of data sometimes requires one or more high-performance computers to make extracting information viable. In the research's area, mainly in public institutions, there is a budget constraint that, in recent years, has shown a growing decrease in available resources, making it impossible to acquire computers that allow such analyses. An alternative to the need for high-performance computers is the use of distributed *computing* techniques. Thus, the orchestration of clustered resources with Kubernetes® was performed for data processing and analysis. In addition, the feasibility of this solution was evaluated in terms of performance, complexity and economic viability, having as a workload the analysis of azithromycin consumption data, in Brazil, between the years 2014 and 2020. Eight computers were used in the structuring the cluster, adding up to 42 CPUs and 88 GiB RAM. For ease of configuration while keeping the *clusters* secure, they were all configured with SSH keys for remote access. The use of *desktops* in the composition of the *cluster* - which will continue to be available for use in the Department of Computer Science - made it possible to process more than 70 GB of data in a distributed manner, in 53 minutes, without any additional cost with the acquisition of equipment for analysis of data. Regarding the data on azithromycin consumption, in Brazil, an increase in prescription rates per 1,000 inhabitants was observed over the time series from 66.2 to 83.8 prescriptions, as well as in almost all federative units, with emphasis on Minas Gerais (66.2 to 149.2 prescriptions per 1,000 inhabitants), Rondônia (37.5 to 109.4 prescriptions per 1,000 inhabitants) and Roraima (37.2 to 99.8 prescriptions per 1,000 inhabitants). A similar behavior was also observed, in the comparative analysis between the years before and during the pandemic (2019 and 2020), with an increase from 59.9 to 83.8 prescriptions per thousand inhabitants, in the country. The accomplishment of this work showed that the use of Kubernetes® *cluster* is a promising alternative for analyzing large volumes of data, especially from the point of view of using underutilized resources, with the recruitment process through dynamic configuration of configuration managers. And that, considering the data analyzed, apparently the COVID-19 pandemic - including the disclosure of azithromycin in COVID kits - may have influenced the greater increase in the consumption of this drug.

Keywords: Kubernetes. Virtualization. Containers. Hypervisor Type 2. Data analysis.

Lista de Figuras

Figura 1 – Evolução do dólar/real na última década	3
Figura 2 – Pagamento efetivo - Ministério da Ciência e Tecnologia	4
Figura 3 – Pagamento efetivo - Relativos aos anos anteriores	4
Figura 4 – Hosted <i>hypervisor</i> diagrama representativo de componentes	9
Figura 5 – Estrutura do <i>container</i>	13
Figura 6 – Eras de <i>deployments</i> e sua evolução por tecnologia base	14
Figura 7 – Kubernetes arquitetura de componentes	15
Figura 8 – Kubernetes Arquitetura de alta disponibilidade	17
Figura 9 – Cronograma geral do trabalho	21
Figura 10 – Ngrok Endpoint	22
Figura 11 – Funcionamento do Ngrok	23
Figura 12 – Fluxo de execução do Ansible Playbook	24
Figura 13 – Airflow - Diagrama de Fluxo	27
Figura 14 – ETL DAG	28
Figura 15 – Airflow - Diagrama de Sequência	28
Figura 16 – S3 armazenamento dos dados	29
Figura 17 – Relatório de execução	30
Figura 18 – Grafico de Gantt da orquestração	31
Figura 19 – Método USE monitoramento de hardware	34
Figura 20 – Monitoramento do <i>cluster</i> pelo Control-Plane	34
Figura 21 – Perfil da aplicação - Utilização dos pods de processamento em relação ao cluster	35
Figura 22 – Precisão por ano	38
Figura 23 – Precisão por ano por estado	38
Figura 24 – Avaliação de correlação de crescimento por ano por estado utilizando tau de Kendall	39

Lista de Tabelas

Tabela 1 – Pagamento efetivo - Ministério da Ciência e Tecnologia	3
Tabela 2 – Termos de pesquisa e estratégia de agrupamento	11
Tabela 3 – Estratégia de pesquisa em IEEE	11
Tabela 4 – Estratégia de pesquisa em <i>Computers and Applied Sciences Complete</i>	12
Tabela 5 – Características das prescrições de azitromicina atendidas em farmácias e drogarias, Brasil, 2014-2020.	33
Tabela 6 – Taxas de prescrição de azitromicina atendidas em farmácias e drogarias, no Brasil, em 2014 e 2020	36
Tabela 7 – Taxas de prescrição de azitromicina atendidas em farmácias e drogarias, no Brasil, em 2019 e 2020	37

Sumário

1 – Introdução	1
1.1 Motivação	1
1.2 Justificativa	1
1.3 Objetivos	4
1.4 Definição e abordagem	5
1.5 Organização do trabalho	6
2 – Fundamentação Teórica	7
2.1 Análise de dados em saúde	7
2.2 Virtualização	8
2.2.1 Virtualização completa	8
2.2.2 Containers ou virtualização em nível de SO	9
2.2.2.1 Container	10
2.2.2.2 Runtime	10
2.2.2.3 Orquestrator	10
2.3 Alternativas open source	10
2.3.1 Termos consultados	10
2.3.2 Seleção de tecnologias	12
2.4 Cluster orquestrador de <i>container</i>	14
3 – Metodologia	16
3.1 Disponibilidade dos recursos deste trabalho	16
3.2 Especificação dos nós integrantes do <i>cluster</i> de baixo custo	16
3.3 Plataforma de orquestração de carga de trabalho	17
3.4 Configuração e provisionamento do <i>cluster</i>	18
3.4.1 Orquestração do processamento de ingestão dos dados	18
3.5 Monitoramento	19
3.6 Análise de dados	20
3.7 Cronograma	20
4 – Análise e Discussão dos Resultados	22
4.1 Provisionamento de infraestrutura	22
4.1.1 Configuração Inicial	22
4.1.2 Configurações dos computadores	23
4.1.3 Configurações de infra do cluster	24
4.1.4 Gestão de armazenamento no <i>cluster</i>	25

4.2	Disponibilização das imagens de container	26
4.3	Orquestração do processamento	27
4.4	Desempenho do cluster	31
4.5	Análise de dados	32
5	– Conclusão	40
5.1	Trabalhos Futuros	41
	Referências	42

1 Introdução

1.1 Motivação

No contexto da análise de dados, diferentes ferramentas estão disponíveis para transformá-los em informação, contudo o uso dessas ferramentas na área da saúde ainda é pouco significativo (GALVÃO; VALENTIM, 2019). Frente a uma tendência crescente de interconexão entre diferentes áreas do conhecimento e do potencial que a análise de dados possibilita para melhoria do sistema de saúde, se faz necessário propor e validar estratégias que permitam o avanço na integração de dados entre diferentes Sistemas de Informação em Saúde (SIS), e que facilitem o processamento e análise do grande volume de dados produzidos e disponibilizados nesses sistemas (GALVÃO; VALENTIM, 2019; MEHTA; PANDIT, 2018a).

Atualmente, conforme determinação do Decreto nº 8.777, de 11 de maio de 2016, que instituiu a Política de Dados Abertos do Poder Executivo Federal (BRASIL, 2016), diversos dados dos SIS são disponibilizados de forma pública. No entanto, apenas a disponibilização dos dados em si não garante que eles poderão ser analisados visando produzir informação relevante para as políticas públicas na área da saúde. É necessário também a disponibilidade de equipes multidisciplinares que garantam a análise desses dados por meio do uso de ferramentas e recursos adequados.

1.2 Justificativa

Alterações de governo e vertentes políticas mudam as prioridades de investimento de recursos públicos, o que faz com que o orçamento de alguns pilares de investimento, a exemplo das áreas de ciência e tecnologia, sofra importantes flutuações ao longo do tempo (Tabela 1). Nos anos de 2018 a 2021, o orçamento destinado à pesquisa e desenvolvimento tecnológico no país sofreu reduções acentuadas, o que tornou ainda mais limitado o acesso a recursos que viabilizem a realização de análise dos dados em ferramentas e infraestruturas tradicionais ou ainda a proposta de novas alternativas.

Em uma avaliação simples desses dados, somado a variação do valor do dólar no mesmo período, pode-se observar (Figura 1) um aumento de mais de 327%. No entanto, o orçamento não acompanhou essa flutuação. Essa ligação, apesar de não ser direta existe, sendo que em ciência e tecnologia equipamentos, licenças de *software*, entre outros recursos utilizados no processamento de dados, são majoritariamente importadas, e portanto em dólar. Como o valor do orçamento efetivo por ano não acompanha o aumento

do dólar, fica claro a perda de poder de compra, dado que a distribuição de verba do orçamento total tenha se mantido.

Algo que valida a flutuação orçamentária é a queda consecutiva de orçamento desde 2018 (Figura 3). Ainda podemos entender que, mesmo depois de 10 anos, temos um orçamento 2,32% menor para ciência e tecnologia. Essa queda na disponibilidade de recursos financeiros impacta diretamente novos projetos, especialmente na compra de equipamentos que viabilizam pesquisas que se valem de bases muito grandes, como as bases de dados do DataSUS e outros sistemas que o compõem. Também pode se exemplificar análises que possuem um maior grau de complexidade como *data linkage* entre bancos de origem distintas como, por exemplo, informações sobre o orçamento da União com informações sobre o pagamento diário e/ou semanal e o banco de despesas do SUS. Sem adequada capacidade computacional à disposição para estudo desses bancos, principalmente em instituições públicas, fica inviável a produção de conhecimento, o monitoramento do emprego dos recursos públicos pela população e qualquer extração de informação viável.

Essas limitações indiretas impõem fortes restrições no processo de avaliação e tomada de decisão embasada em dados. Ainda avaliando esse impacto econômico do problema descrito acima, é preciso pensar, como já discutido, a complexidade da tomada de decisões em saúde. Sem informações que embasem essas decisões, fica restrita a capacidade de compreensão do cenário nacional, sobre a situação de saúde pública. Quando leva-se em consideração que o SUS é um sistema subfinanciado, quando comparado a países que também possuem sistemas de saúde pública e gratuitas. O valor do orçamento por dia por pessoa é de apenas R\$3,83 o que é um preceito quando considera-se restrição orçamentária do SUS por pessoa. Para entender a complexidade de um sistema tão abrangente é preciso avaliar sua escala. Mais de 152 milhões de pessoas utilizam exclusivamente o SUS como promotor de saúde.

Assim, considerando os impactos econômicos e sociais apresentados, o presente trabalho se propõe a discutir alternativas e abordagens que amenizem impactos de orçamento disponibilizados pelo governo, para a viabilização de projetos de pesquisa que utilizam bases de dados tão grandes.

Nesse sentido, temos dois efeitos diretos:

- Redução de fragilidade orçamentária; e
- Proposta de alternativas para análise de dados em saúde.

A redução de fragilidade orçamentária se dá pela proposta de utilização de recursos já disponíveis e subutilizados, como *desktops* de bibliotecas, laboratórios etc, por meio de ambientes virtualizados, compartilhando esses recursos com a orquestração das cargas de trabalhos de análise de dados. Para viabilizar essa proposta, o trabalho compara a perfor-

Tabela 1 – Pagamento efetivo - Ministério da Ciência e Tecnologia

Ano	Objetivo
2010	R\$ 6.288.931.123,00
2011	R\$ 5.918.584.706,00
2012	R\$ 6.918.288.201,00
2013	R\$ 7.787.464.592,00
2014	R\$ 8.598.785.224,00
2015	R\$ 7.964.319.815,00
2016	R\$ 8.404.014.691,00
2017	R\$ 9.085.620.227,00
2018	R\$ 9.157.748.260,00
2019	R\$ 8.812.096.752,00
2020	R\$ 7.859.851.948,00
2021	R\$ 6.142.873.884,00

Fonte: SIOP consulta realizada em Janeiro de 2022



Figura 1 – Evolução do dólar/real na última década (Fonte: Feito pelo autor)

mance de sistemas virtuais em computadores com baixo poder computacional, propondo assim uma forma de abordar, tanto o isolamento, para processos mais sensíveis, quanto a utilização de recursos que já estão disponíveis, a fim de garantir o menor CAPEX possível para excussão do projeto de pesquisa.

Para avaliação dessa alternativa toma-se como exemplo os bancos de “Vendas de Medicamentos Controlados e Antimicrobianos - Medicamentos Industrializados”, objeto de análise deste trabalho. Nesse banco estão disponíveis cerca de 70 GB e mais de 500 milhões de linhas de dados sobre a comercialização de medicamentos de venda controlada no país. Logo, tão importante quanto a disponibilidade pública dos dados, é fundamental encontrar estratégias técnicas e economicamente viáveis a fim de possibilitar

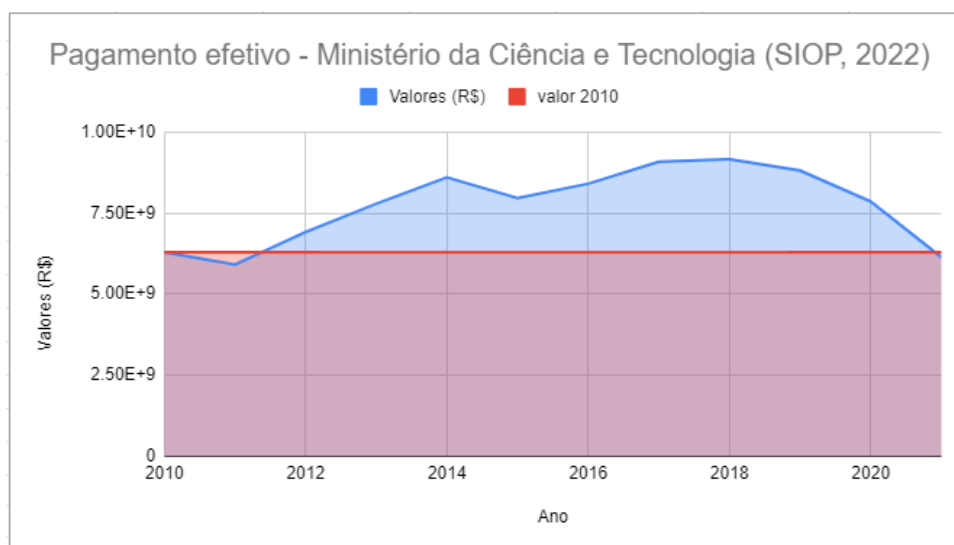


Figura 2 – Pagamento efetivo - Ministério da Ciência e Tecnologia (Fonte: Feito pelo autor)

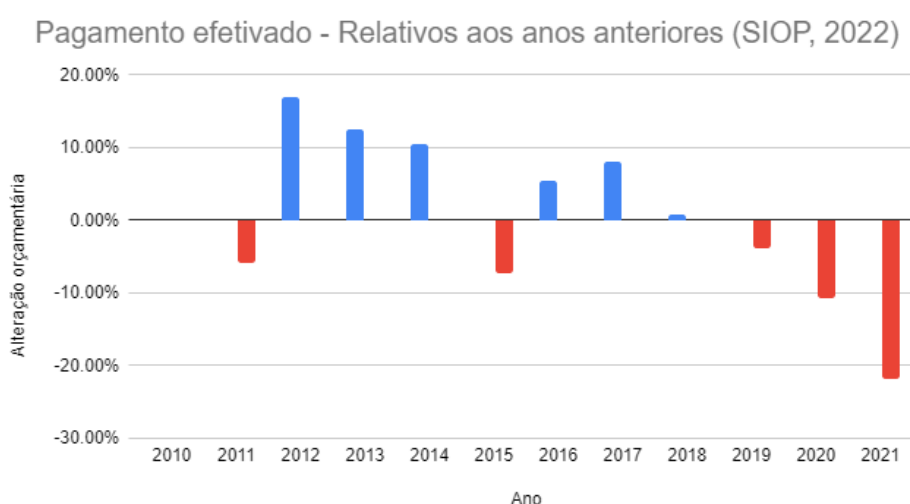


Figura 3 – Pagamento efetivo - Relativos aos anos anteriores (Fonte: Feito pelo autor)

que pesquisadores em todo o país possam contribuir com a análise e a interpretação desses dados, mesmo frente a baixa disponibilidade de recursos financeiros e de infraestrutura, como servidores de alta performance (HPC), por exemplo.

1.3 Objetivos

Diante disso, com a realização deste trabalho espera-se oferecer uma alternativa para análise de grandes volumes de dados que possua baixo custo financeiro, menor complexidade de configuração, maior efetividade (menor tempo de análise) e que não seja dependente da disponibilidade ou uso de recursos dedicados, como computadores de alta performance (HPC). A abordagem que irá se seguir é na utilização de alternativas *open*

source atualmente disponíveis e que sejam possíveis de serem utilizadas em computadores de baixo custo e menor poder computacional - ex.: *OpenStack*, *CloudStack* etc.

Deste modo, espera-se demonstrar comparativamente a implementação de uma solução para análise de dados em plataformas de orquestração de *containers*, que permita recrutar computadores comuns para essa análise. E assim, espera-se, superar de maneira custo-efetiva um problema de restrição orçamentária e técnica para instituições públicas e grupos de pesquisa que realizam análises de grande volumes de dados. No caso deste trabalho, a aplicação está direcionada para a área da saúde, utilizando uma tecnologia já amplamente empregada no setor privado, o que viabiliza o suporte de estudantes e/ou profissionais das áreas de Engenharias e Computação. Espera-se, ainda, contribuir para que os dados públicos em saúde sejam analisados com maior frequência e menor restrição, gerando indicadores melhores e atualizados para melhor tomada de decisão em saúde.

1.4 Definição e abordagem

A proposta do trabalho visa avaliar a utilização de *cluster* de Kubernetes® como plataforma de orquestração de cargas de trabalho para processamento de dados em *clusters* compostos por computadores de baixo custo ou reaproveitados. Utilizando conceitos e ferramentas DevOps (BASS et al, 2015) intenta-se abordar o provisionamento, a integração e o *deploy* da infra estrutura, valendo-se da automação em esteiras de CI (continuous integration), CD (continuous delivery). Utiliza-se também de ferramentas de IaC (Infrastructure as Code), que tornam a configuração e disponibilização desse *cluster* mais ágil, auto documentada e diminui a necessidade de operação e manutenções no *cluster*, outros recursos e aplicações configuradas para a solução proposta nesse trabalho.

A utilização da plataforma visa validar seu uso para orquestração de tarefas em paralelo e processamento distribuído de dados durante a análise, permitindo o uso simultâneo de diversas máquinas. Inicialmente, foram utilizados 8 computadores com capacidades de processamento semelhantes a computadores *desktop* de 8-16 GB (Gigabytes) de RAM (Random Access Memory) e 4-8 vCPU (virtual Central Process Unit). Essa restrição permitiu uma análise de viabilidade para que o processamento e análise de grandes massas de dados (maiores que 50 GB) possam ser feitas sem o uso de HPC.

Utilizando como carga de trabalho a análise de tendência de consumo de azitromicina no Brasil, entre os anos de 2014 e 2021. Pretende-se utilizar, como principal resultado, a viabilidade de utilização de computadores do tipo *desktop* para orquestração e análise de grandes massas de dados como dados do SUS (Sistema Único de Saúde) garantindo assim viabilidade de trabalhos e proponto a utilização dessa plataforma em computadores menos específicos, como alternativa a HPC.

O trabalho não engloba a realização de interpretação da informação gerada pelo

banco, garantindo, assim, apenas o resultado correto da análise citada como carga de trabalho para comparação. Também não está sendo proposta uma metodologia de análise do banco referenciado, mas a avaliação das tecnologias empregadas para orquestração das tarefas, comparação de desempenho entre as alternativas da implementação da plataforma e sua implementação como proposta para uso mais amplo nas instituições sob restrição orçamentária, com o fim de continuar a realizar análises de dados, ainda que sem hardware adequado.

As principais métricas visadas para avaliar a viabilidade do *cluster* são tempo de execução da atividade de importação e processamento do banco, taxa de utilização de CPU e memória do *cluster*, garantido que o mesmo permaneça disponível para outras cargas de trabalho, i.e não ocupando toda a capacidade disponível, para que ainda possam ser executadas atividades simultâneas no *cluster* não relacionadas a análise. Métricas complementares, com taxa de leitura e escrita em disco e tráfego de rede são coletadas e discutidas, mas não são o principal objetivo desse trabalho, sendo que não considera-se aqui comparação com diferentes tipos de protocolos para armazenamento. Sendo que protocolos diferentes de armazenamento compartilhado ou distribuídos em redes podem afetar diretamente na taxa de utilização da rede.

1.5 Organização do trabalho

Este trabalho está dividido em cinco capítulos. Na Introdução foi apresentado o contexto geral do trabalho, tecnologias que são avaliadas e possíveis problemáticas econômicas e sociais.

O Capítulo 2 apresenta a fundamentação teórica e a revisão da literatura, discorrendo sobre as leituras que embasaram toda a construção do projeto. Nele também são aprofundados em conceitos necessários para a discussão e condução do trabalho.

O Capítulo 3 apresenta a metodologia utilizada para a construção dos componentes do projeto e a forma de avaliação de desempenho dos ambientes propostos, bem como a forma de avaliação.

O Capítulo 4 apresenta os resultados obtidos tanto para orquestração das cargas de trabalho no ETL (extração, transformação e carga) e também os resultados obtidos para a análise e processamento dos dados de consumo e prescrição obtidos para azitromicina, no período descrito anteriormente. São discutidos os resultados obtidos para a orquestração do *cluster* e na análise dos dados de prescrição da azitromicina, além de suas interpretações.

O Capítulo 5 apresenta as conclusões do trabalho, incluindo o que o aluno realizou ao longo da execução do trabalho, se a proposta do trabalho atingiu a expectativa como solução ao problema proposto, além de possibilidades para trabalhos futuros.

2 Fundamentação Teórica

2.1 Análise de dados em saúde

A tomada de decisões em saúde precisa com frequência do suporte de um profissional especializado na temática a ser decidida. Uma série de técnicas são aplicadas por esse profissional para avaliação do contexto, uma vez que os dados puros não são suficientes, devido à multifatorialidade (ANDRADE, 2008; RESENDE; VIANA; VIDGAL, 2009). O grande número de sistemas de auxílio em saúde, sejam esses de ordem regulatória ou ainda por iniciativas privadas, fazem com que o volume de dados cresça exponencialmente, produzindo o fenômeno de *Big Data*. Uma definição conhecida desse conceito aponta volume, variedade e velocidade como os três vetores relacionados à produção massiva de dados (LANEY et al., 2001). Outras características foram adicionadas a essa primeira definição de *Big Data*, tais como veracidade (SCHROECK et al., 2012), complexidade e desestruturação (CORPORATION, 2012), e valor (DIJCKS, 2013).

A multifatorialidade já citada é um aspecto que se relaciona à complexidade dos dados em grande volumes. É cada vez mais difícil estabelecer uma relação de causa e efeito para a tomada de decisão segura, sem auxílio ou sumarização desses dados em informações mais tangíveis e associadas. Também é difícil descrever um algoritmo que permita analisar todos os dados de contexto e relacioná-los de forma que possam ser utilizados como substituição ao conhecimento tácito de um profissional da saúde experiente (FACELI et al., 2011).

O suporte de sistemas, métodos e práticas que auxiliem a assistência em saúde no tratamento de *Big Data* ainda não é suficientemente consolidado. Faltam evidências de aplicações, especialmente quantitativas, que validem viabilidade e eficácia na utilização de ferramentas e técnicas no tratamento de *Big Data* no campo de conhecimento da saúde. Além disso, boa parte das tecnologias utilizadas nesses estudos, utilizam recursos específicos de computação e, frequentemente, esses recursos têm custo elevado. Esses estudos focam na avaliação de dados para otimização de recursos, suporte a decisões clínicas e redução de custo do cuidado (MEHTA; PANDIT, 2018b) e não na análise de dados para tomadas de decisão em saúde pública, nem tão pouco no auxílio para construção de políticas públicas. Ambos os focos são válidos, porém apresentam conceitos distintos e a forma de extrair essas informações são diferentes.

Há uma grande oportunidade de estudo na proposição de um *stacks* (conjunto de ferramentas) e métodos para extrair essas informações no contexto de saúde pública. O desenvolvimento de abordagens para o processamento de dados nesse contexto de

estudo pode viabilizar a utilização de grandes bases de dados públicas o que permitiria o estabelecimento de novas correlações e, posteriormente, a associação com demais bancos ou conjuntos de informações.

Sendo o paragrafo acima, justamente o que esse trabalho se propõe, ainda que o volume de dados da base sob análise neste trabalho e sua composição não qualifiquem como *Big Data*, sob a perspectiva de LANEY. Esse trabalho pode ser utilizado de base para uma nova forma de utilização de recursos computacionais na análise de grandes massas de dados e a posteriori *Big Data*.

2.2 Virtualização

Virtualização, no contexto de computação, refere-se à abstração de componentes e/ou recursos de computador. O objetivo do ambiente de computação virtual é melhorar a utilização de recursos, por meio da utilização sob demanda destes recursos, sendo essa demanda a forma de utilização e a quantidade de recursos necessários. Virtualização fornece uma plataforma operacional unificada e integrada para usuários e aplicativos, com base na agregação de recursos heterogêneos e autônomos.

Essa abstração é, usualmente, feita por uma camada de software entre *hardware* e SO (SO), chamada de *hypervisor* ou monitor de máquina virtual (VMM).

Mais recentemente, a virtualização em todos os níveis (sistema, armazenamento e rede) tornou-se importante novamente como forma de melhorar a segurança, confiabilidade e disponibilidade do sistema, reduzir custos e fornecer maior flexibilidade. (SAHOO; MOHAPATRA; LATH, 2010)

Existem diversos níveis de abstração para virtualização, considerando níveis mais detalhados de funcionalidades e/ou níveis menos específicos de hardware. Isso torna possível extrair aplicações e seus respectivos ambientes com mais facilidade e trocá-los de máquina base, considerando essa mudança como portabilidade. Como objeto de estudo deste trabalho nos limitamos a definição de dois tipos de virtualização: completa, nível de SO.

2.2.1 Virtualização completa

Virtualização completa, sendo intermediada por sistemas VMM (gerenciador de máquina virtual) ou *hypervisor*, é executado em cima de um SO *host*, geralmente como um aplicativo no SO base. O resultado é que, nas VMs, os aplicativos e o SO convidado são executados em cima de um hardware virtual fornecido pelo *hypervisor*. Pode ser considerado para fornecer "Virtualização Completa".

Neste tipo de configuração, os dispositivos de E/S são atribuídos às máquinas

convidadas (*guest*), imitando os dispositivos físicos. A interação com esses dispositivos no ambiente virtual são então direcionados para os dispositivos físicos reais, seja pelo *driver* do SO do host ou pelo "*Driver VM*".

Essa arquitetura pode ser observada na Figura 4. A principal vantagem dessa abordagem é que é muito fácil de usar. Um usuário comum pode instalar um produto de software como o @VirtualBox como qualquer outro produto de software no SO de sua escolha. Dentro do deste virtualizador, um SO *guest* pode ser instalado e usado como se estivesse sendo executado diretamente no hardware. (PORTNOY, 2012)

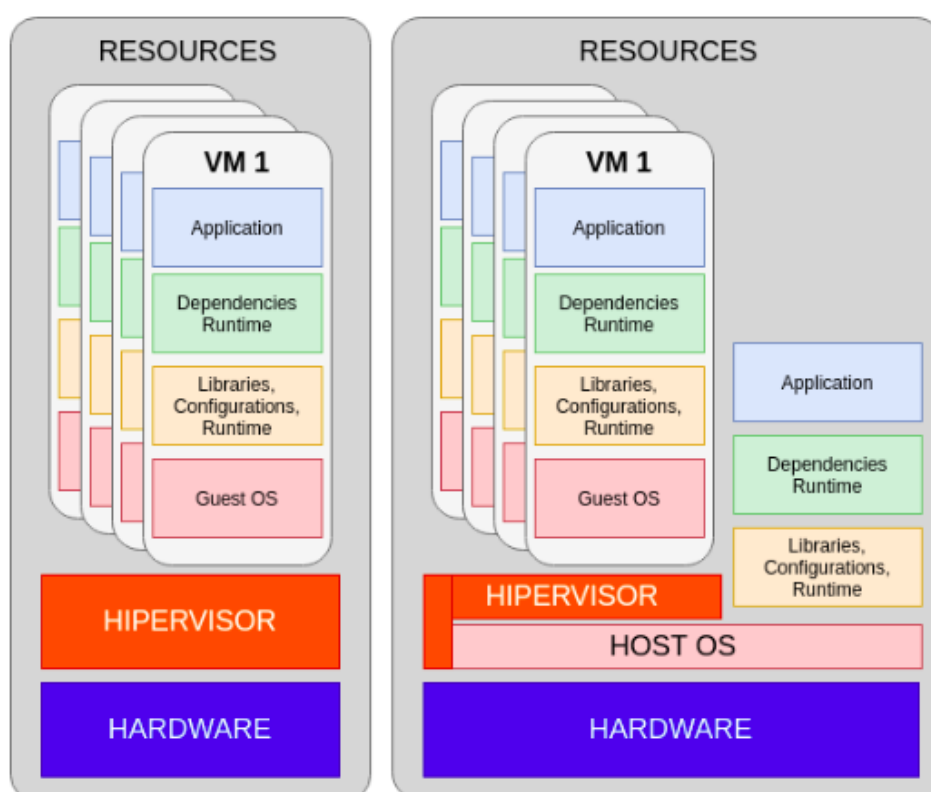


Figura 4 – Hosted *hypervisor* diagrama representativo de componentes (Feita pelo autor)

2.2.2 Containers ou virtualização em nível de SO

Conhecida também como *container*, a virtualização em nível SO é uma técnica de virtualização que referencia ao uso compartilhado do kernel do SO, porém manipula a forma de execução dos processos no *userspace*, setando dois conjuntos de configurações, *namespace* (isolamento dos recursos de sistema) e *cgroup* (controle de utilização dos recursos) (MACCARTY, 2018).

2.2.2.1 Container

Containers são um conjunto de um ou mais processos sendo executados isolados do sistema base.

2.2.2.2 Runtime

Como no *hypervisor* que executa e comunica com o sistema base ou hardware, para *container* temos o *container runtime* que é responsável por executar o *container* e gerenciar o namespace e gcgroups definido para aquele *container*.

2.2.2.3 Orquestrator

O orquestrador de *container* é o sistema responsável por avaliar e comandar a execução dos *container* por meio do *runtime*, tendo como premissa a disponibilização dos recursos totais da máquina e garantindo que o *container* seja executado com uma serie de parâmetros, bem como também é responsável pelas configurações de rede que são utilizadas e gerenciadas ao longo do ciclo de vida dos *containers* sob sua orquestração.

2.3 Alternativas open source

Para avaliar alternativas no mercado a seguinte estratégia de pesquisa foi conduzida nas seguintes bases:

- IEEE Xplore
- *Computers and Applied Sciences Complete*

2.3.1 Termos consultados

Para organizar os termos selecionados para a busca, bem como a suas variações de consulta foi realizado um agrupamento de palavras em temas e utilizadas combinações de termos de cada tema, segundo a estrutura apresentada na Tabela 2.

A formatação da estratégia de pesquisa está contida nas Tabelas 3 e 4. Essa formatação permite indicar estudos que tenham os temas indicados de forma reduzida por assunto, título, palavras-chave e resumo e, a partir dessa extração, uma avaliação de ferramentas utilizadas resultou no levantamento de tecnologias a serem avaliadas na próxima sessão. Levando em consideração um filtro adicional que é ser *Open Source*.

CLUSTER	DATA PROCESSING	COST	COMPUTING
cluster "small cluster"	"data process" "data processing" "data processing system"	cost "low cost" "lowest cost" "cheap" "economic" "inexpensive"	computing "grid computing" "mist computing" "cloud computing" "fog computing" "edge computing"
#1 Combinar OR	#2 Combinar OR	#4 Combinar OR	#6 Combinar com OR
	#3 #1 AND #2	#5 #3 AND #4	#7 #5 AND #6

Tabela 2 – Termos de pesquisa e estratégia de agrupamento

Passo	Operação	Resultados
#1	ALL = (cluster OR "small cluster")	1.199.979
#2	ALL = ("data process" OR "data processing" OR "data processing system")	88.988
#3	#1 AND #2	5.810
#4	ALL = ("low cost" OR "lowest cost" OR cheap OR economic OR inexpensive)	1.836.782
#5	#3 AND #4	206
#6	ALL = ("grid computing" OR "mist computing" OR "cloud computing" OR "fog computing" OR "edge computing")	81.679
#7	#5 AND #6	22

Tabela 3 – Estratégia de pesquisa em IEEE

Passo	Operação	Resultados
#1	TITLE-ABS-KEY (cluster OR "small cluster")	963.442
#2	TITLE-ABS-KEY ("data process" OR "data processing" OR "data processing system")	316.839
#3	#1 AND #2	8.190
#4	TITLE-ABS-KEY ("low cost" OR "lowest cost" OR cheap OR economic OR inexpensive)	2.294.081
#5	#3 AND #4	347
#6	ALL = ("grid computing" OR "mist computing" OR "cloud computing" OR "fog computing" OR "edge computing")	125.105
#7	#5 AND #6	

Tabela 4 – Estratégia de pesquisa em *Computers and Applied Sciences Complete*

2.3.2 Seleção de tecnologias

Com a pesquisa descrita acima, pudemos filtrar tecnologias e ferramentas de mercado utilizadas em *clusters* de baixo custo e como estratégias para a análise de dados, o que retornou um grande número de artigos. Para fins de apresentação dessas soluções neste trabalho, elas foram agrupadas conforme níveis de complexidade de implementação, sempre mantendo o custo como restrição primária e o contexto de aplicação sendo para instituições com grande restrição orçamentária. Diante disso, e tendo em vista o tamanho da base de dados de origem, durante o levantamento de material para o projeto foram avaliadas soluções em dois grupos:

- Computação em nuvem privada; e
- Orquestração de *Containers*.

Na primeira categoria, a utilização de ferramentas como *OpenStack®* e *CloudStack®* foram consideradas. No entanto, observa-se que nesse tipo de utilização existe um *overhead* substancial, tanto em termos de complexidade de configuração, quanto em termos de

hardware necessário para operar de maneira eficiente. Essas alternativas requerem maior capacidade computacional, conforme suas configurações recomendadas (CLOUDSTACK, 2022; OPENSTACK, 2022). Além disso, as soluções de nuvem privada estendem muito o propósito de orquestração de cargas de trabalho, provendo todo o conceito de infraestrutura como serviço (IaaS). E, dependendo da implementação e dos componentes utilizados, elas provêm plataforma como serviço (PaaS), que são categorias de abstração do hardware, configurações e sistemas de suporte/apoio como sistema operacional (SO) para aplicação (OPENSTACK, 2022; CLOUDSTACK, 2022; MELL; GRANCE, 2011). Logo, embora sejam alternativas populares, tornam o processo substancialmente mais complexo e, portanto, foram descartadas para essa avaliação, tendo em vista o escopo deste trabalho.

Na segunda categoria, sistemas de orquestração de *containers* possuem um baixo *overhead* devido à arquitetura do *container* (Figura 5), compartilhando parte do kernel space. Logo, não necessitam de uma camada de virtualização do SO, presente em virtualizações completas e gerenciadas por *hypervisors* (Figura 6). Vale ressaltar que, essa solução oferece ainda algumas possibilidades como, por exemplo, o gerenciamento de capacidade dos nós do *cluster* para agendamento de tarefas. Dentre as plataformas disponíveis, o Kubernetes® é apontado como uma das principais soluções de orquestração de *containers*, tanto pela disponibilidade de features, quando pelos projetos em operação e pelo tamanho de sua comunidade (TRUYEN et al., 2021). O Kubernetes® tem o apoio de entidades como Cloud Native Computing Foundation (CNCF), que apoiam e supervisionam a plataforma de software - definida como peça importante de software, sob o qual diversos programas de aplicativos menores podem ser projetados para serem executados (PLATFORM..., 2022). Esse apoio tem como objetivo a expansão das capacidades, endereçando problemas conhecidos e situações de uso, bem como estabelecendo padrões para tecnologias que pertencem ao ecossistema de orquestração de *containers*.

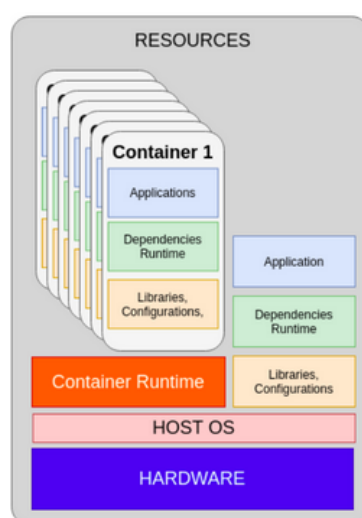


Figura 5 – Estrutura do *container* (elaborada pelo autor)

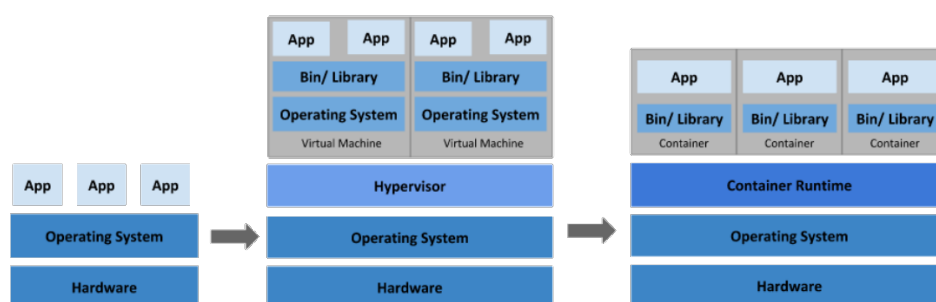


Figura 6 – Eras de *deployments* e sua evolução por tecnologia base (Fonte: The Linux Foundation®, 2021)

Soluções como Apache Mesos, Hashicorp Nomad, e Docker Swarm também foram avaliadas em estudos (TRUYEN et al., 2021), mas em todos os casos foram citadas diferenças significativas, especialmente no uso, sendo o Kubernetes® a melhor por eles avaliada.

2.4 Cluster orquestrador de *container*

Kubernetes® é a consolidação de quinze anos de trabalho da Google® com orquestração de cargas de trabalho, processamentos *batch*, e um sistema interno de gerenciamento de *cluster* orientado a *containers*, o Borg (VERMA et al., 2015).

As estruturas básicas do Kubernetes® são divididas em componentes com atribuições bem definidas, como na Figura 7. Os componentes que são essenciais à proposta deste trabalho são:

- *Kube-apiserver*, que concentra toda a api do Kubernetes®
- *Kube-scheduler*, que avalia novos pods e em qual nó *worker* do *cluster* eles são alocados
- *Kube-controller-manager*, que comporta os objetos de controle do Kubernetes®
- *Kubelet*, responsável por repassar comando do control plane, para o *worker* e comunicar com *container runtime*
- *Kube-proxy*, responsável por toda a estrutura de redes nativa do Kubernetes® e pods do *cluster*
- *Pod*, menor unidade de *deployment*, podendo conter um conjunto de *containers* que compõem uma solução.

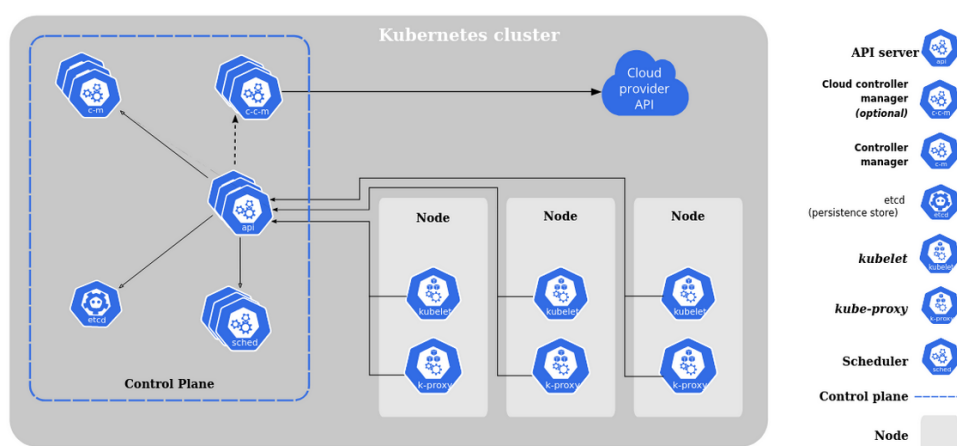


Figura 7 – Kubernetes arquitetura de componentes (Fonte: The Linux Foundation®, 2021)

3 Metodologia

3.1 Disponibilidade dos recursos deste trabalho

Todos os componentes definidos nesse trabalho estão em um repositório público no Github: <https://github.com/felipefrocha/esufmg-tcc>, garantindo a livre apreciação da comunidade não só científica, mas a todos os interessados na contribuição ou utilização sob a licença pública geral GNU versão 3 (GNU, 2022).

3.2 Especificação dos nós integrantes do *cluster* de baixo custo

A formação do *cluster* é de máquinas físicas que possuem custo e capacidade computacionais mais baixos, com configurações comumente encontradas em computadores do tipo *desktops*, como os utilizados em residências, escritórios e laboratórios de informática genéricos.

A primeira versão dessa solução foi realizada de maneira simulada, utilizando ambiente virtualizado com VMs que foram provisionadas em *hypervisors* do tipo 2 (COMER, 2021) (*hosted hypervisor*), para garantir a simplicidade da implementação dos conceitos abordados no trabalho e validar a abordagem de provisionamento, configuração e *deploy* de aplicações testes e laboratório das ferramentas de monitoramento que são utilizadas, a serem descritas na sessão 3.5.

O desenvolvimento desse trabalho é o provisionamento do *cluster* nos ambientes de teste configurados como descrito na sessão 1.4, sendo que algumas premissas são utilizadas, como o uso de mesmo sistema operacional e versão em cada um desses computadores, garantindo a redução de configuração necessária para a implementação. Outra premissa utilizada para esse estudo é a utilização de uma rede comum aos computadores.

O aproveitamento de computadores comuns já existentes e subutilizados, seja por uso abaixo de sua capacidade ou ainda intervalos de ociosidade, qualifica o baixo custo da formação do *cluster* em questão, apresentando CAPEX (capital expenditure), ou investimento inicial mínimo, se não zero.

3.3 Plataforma de orquestração de carga de trabalho

A plataforma de *cluster* e orquestração de cargas de trabalho utilizadas nesse trabalho é o Kubernetes. A arquitetura de implementação do *cluster* é de *multi-master* com etcd (ETCD, 2022) (controlador de logs, e estado do *cluster*) anexado (KUBERNETES, 2022), ou rodando nos mesmos computadores masters do *cluster*. A arquitetura apresentada garante (Figura 8) alta disponibilidade do *cluster*, importante para que não haja interrupções inesperadas durante a orquestração das cargas de trabalho (execução, disponibilidade e garantia de estado desejado). Ela apenas não garante uma recuperação rápida de quedas dos servidores dos nós (computador/servidor integrante do *cluster*) mestres, significando a perda de todos os estados e, com isso, havendo a necessidade de reconfiguração do *cluster*. Porém, considerando os recursos limitados, na justificativa desse projeto, a alta disponibilidade de estado, significaria na obrigatoriedade do mesmo número de computadores disponibilizados para etcd e masters para controle dos estados.

kubeadm HA topology - stacked etcd

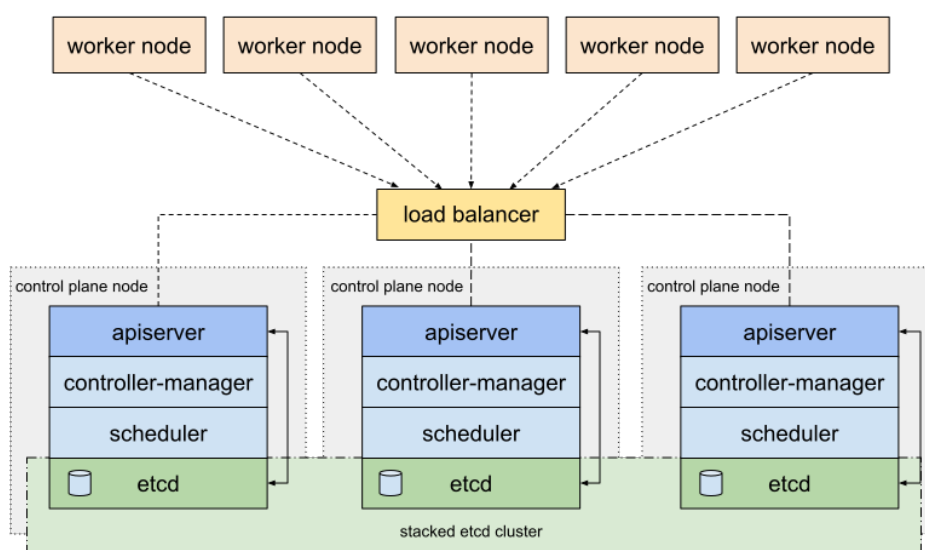


Figura 8 – Kubernetes Arquitetura de alta disponibilidade (Fonte: The Linux Foundation®, 2021)

Toda a implementação da solução e os componentes relacionados estão containerizados, possibilitando sua orquestração pelo *cluster* de Kubernetes®. Apenas as configurações dos *clusters* e seu provisionamento não são containerizados. Esses estão disponibilizados em outro repositório específico. Para ciclo de vida da aplicação e configurações gerais da solução é utilizada uma estratégia de organização de código em monorepo, sendo essa estratégia a utilização de um repositório único para acompanhamento e desenvolvimento de todos os componentes de software e configurações. Isso facilita a visualização,

centralização, sincronização e padronização como benefícios primários, conforme citado na literatura, reforçando a adoção dessa estratégia (BRITO; TERRA; VALENTE, 2018).

3.4 Configuração e provisionamento do *cluster*

Oito computadores foram utilizados na estruturação do cluster, somando 42 núcleos e 88 GiB RAM. Neles foi, inicialmente, instalada a versão LTS (*long term support*) da distribuição de Linux® Ubuntu® Server 20.04.3. Para facilidade de configuração, mantendo a segurança dos *clusters*, todos foram configurados com chaves SSH para acesso remoto. O laboratório Winet do DCC UFMG, parceiro na realização desse trabalho, cedeu uma de suas sub-redes de comprimento /25 (CIDR, *Classless Inter-Domain Routing*).

Para provisionamento do *cluster* é utilizado Ansible®, da empresa RedHat® que é um sistema de gerenciamento de configuração (CMS). Sua adoção se deu pela característica minimalista de configuração inicial, facilidade de uso e uma característica fundamental que diminui o *overhead* de operação necessária para sua utilização: não possuir agente instalado no inventário de máquinas gerenciadas.

O principal ganho na utilização de CMS é a manutenção e replicabilidade de uma determinada configuração e, no caso do Ansible, não há necessidade de configuração prévia ou instalação de nenhum binário específico para sua utilização, reduzindo assim a complexidade de sua adoção.

A configuração inicial é realizada pela disponibilidade de: acesso via rede e python instalado na máquina. O gerenciamento dessa configuração é feita por ativo (asset, ou recurso) relacionado em uma lista de recursos agrupadas, chamada inventário. Também é possível utilizar alguns tipos de autenticação (kerberos, WinRM, SSH etc). O tipo utilizado no trabalho foi o protocolo SSH (RFC4254, 2006) por chaves assimétricas, o que garante um nível aceitável de segurança, especialmente quando é possível escolher os algoritmos de criptografia e suas possíveis variações como RSA e ED25519.

3.4.1 Orquestração do processamento de ingestão dos dados

Para obtenção e trabalho com os dados propostos, escolheu-se trabalhar com um fluxo conhecido de mercado, aplicado tanto em rotinas de batelada (*batch*), como em processos de contínuos *stream*. A extração, transformação e carga (ETL) dos dados foi obtida de uma fonte de armazenamento remota por requisição *download*, via endereço de URL (*Uniform Resource Locators*), seguido da transformação (filtragem, sumarização) e carregado em um banco de dados relacional localizado no *cluster*, no caso PostgreSQL®.

A orquestração das *tasks* (tarefas), no conceito apresentado pela ferramenta Apache Airflow® (AIRFLOW, 2022), permite realizar a execução em uma ordem específica, essa

estrutura é definida em uma DAG(*direct acyclic graph*). Em execuções simultâneas e paralelas de tarefas, utilizou-se o modo de execução associado a API (*application programming interface*) do Kubernetes®. Dessa forma, cada *task* é criada em um novo *pod* com recursos definidos em sua especificação *task* durante o paralelismo dinâmico dos processos(*fanout pattern*).

Essa técnica é comumente associada a padrões de execução *serverless*, usualmente utilizando um processo de subscrição por mensagem dos novos processos. No caso do mapeamento dinâmico das tarefas orquestradas pelo Airflow, uma vez que são utilizados pods independentes por tarefa, podemos estrapolar o termo. Para melhor entendimento pode-se fazer uma comparação com *fork* de processos em processamento paralelo, salvo que no contexto do cluster, o pod (unidade onde a tarefa é executada) pode ou não ser instanciado no mesmo computador de origem do executor, e por meio de *callback* com o *scheduler* do *cluster* - uma comunicação por um banco de metadados -, o executor gera os pods de execução das tarefas, controla o estado da tarefa e seu retorno.

3.5 Monitoramento

São utilizados para monitoramento de execução das cargas de trabalho o Prometheus® e para visualização dos dados Grafana®, ambos sendo configurados a partir do provisionamento do *cluster*, ainda com a ferramenta proposta inicialmente Ansible®. Dessa forma, é possível avaliar parâmetros de taxa de lotação das máquinas base, pelo parâmetro de processador e memória, operações de leitura e escrita no disco e também tráfego de rede. Por meio dessas ferramentas é possível, ainda, avaliar dados de tempo de execução de cargas de trabalho em ambos os ambientes propostos e, assim, poder compará-los quanto a eficiência de uso de hardware.

O tipo de teste de aplicação é um macrobenchmark (system level benchmark) (HUGE, 2022; SCHEEPERS, 2014) comparando parâmetros de uso de CPU, memória e tempo de execução da carga de trabalho proposta na sessão 3.5 deste trabalho. Os parâmetros são avaliados tanto na máquina de suporte ao container, como também os containers em si, bem como o tempo de provisionamento do *container* da carga de trabalho, tempo de execução e quantidades de falhas.

Esse método USE (Usage, Saturation and Errors) (GREGG, 2022) é utilizado para extrair e apresentar métricas e avaliar possíveis problemas. Destaca-se que esse não é o foco do trabalho, mas captura de forma adequada os parâmetros de *hardware* citados acima.

O controle de tempo é através de ferramentas de orquestração Apache Airflow®. O que permite visualizar detalhadamente o tempo de execução, início e finalização das tarefas.

3.6 Análise de dados

O banco de dados “Vendas de Medicamentos Controlados e Antimicrobianos - Medicamentos Industrializados”, disponibilizado pelo governo brasileiro (via portal dados.gov.br), é utilizado nesse trabalho. Os anos correspondentes dos dados são no período entre 2014 e 2020 e o banco possui mais de 70 GB e mais de 530 milhões de linhas sendo, portanto, suficiente para ser utilizado como carga de trabalho.

Um banco de dados contendo todas as prescrições de azitromicina dispensadas por farmácias privadas e drogarias brasileiras, no período de 2014 a 2020, é construído a partir dos dados processados dos bancos de “Vendas de Medicamentos Controlados e Antimicrobianos - Medicamentos Industrializados”. A azitromicina foi escolhida como objeto de análise de consumo para os períodos pré e durante a pandemia da COVID-19, por se tratar do antimicrobiano – sujeito à venda controlada no Brasil – mas amplamente indicado para a prevenção ou tratamento da COVID-19, mesmo sem comprovação de eficácia (SANTOS-PINTO; MIRANDA; CASTRO, 2021).

As seguintes variáveis são coletadas: apresentação, quantidade dispensada e unidade federativa (UF) de comercialização de cada medicamento; idade e sexo do paciente. São mensuradas as tendências de consumo por meio do número unidades dispensadas – caixas ou frascos –, avaliadas usando o índice de correlação tau de Kendall para a quantidade de medicamento dispensado ao longo dos anos de 2014 a 2020. As mudanças no tamanho da população no tempo são consideradas calculando a taxa de prescrição de azitromicina atendida por 1000 pessoas, tendo como referência estimativas anuais da população (denominador), obtidas a partir de dados do Instituto Brasileiro de Geografia e Estatística (IBGE). Também são comparados os consumos da azitromicina no período anterior a pandemia (2019) e durante a pandemia (2020).

3.7 Cronograma

Na Figura 9 é apresentado o cronograma completo do projeto.

Fases do Projeto	Atividades do TCC	Ondas		Dates		Cascata	
		Atividades	Objetivos	S	Data Inicial	Data Final	
Exploratório	Proposta TCC I	Elaboração de estratégias de busca	Identificar estudos parecidos, explorar tecnologias disponíveis e avaliar oportunidades e conceitos associados aos usuários	1	17/10/2021	23/10/2021	
		Busca e avaliação dos artigos selecionados		2	24/10/2021	30/10/2021	
		Escrita de revisão bibliográfica		3	31/10/2021	06/11/2021	
Concepção	Visão Geral do Projeto	Descrição formal dos stakeholders	Identificar público alvo, validar ideia da solução e listar alternativas	7	07/11/2021	04/12/2021	
		Avaliação de alternativas		9	05/12/2021	18/12/2021	
		Elaboração da fundamentação teórica e justificativa		10	19/12/2021	25/12/2021	
Desenvolvimento	Matrizes da Defesa Texto Inicial Monografia & Versão Final da Monografia	Especificação e critérios de aceitação	Elaborar detalhamento da solução, mapear fronteiras da solução, identificar riscos ao projeto e propor desenho inicial da solução	12	26/12/2021	08/01/2022	
		Levantamento de Requisitos		13	09/01/2022	15/01/2022	
		Levantamento de Lista de Materiais e softwares		15	16/01/2022	29/01/2022	
		Apresentação do estudo e resultados de PoCs		17	30/01/2022	12/02/2022	
		Avaliação de viabilidade do sistema		20	13/02/2022	05/03/2022	
Produção		Implementação da montagem (caso viável) e testes de verificação	Produção, Inspeção, Verificação e Validação da solução proposta.	24	06/03/2022	02/04/2022	
		Instrumentação (software) e verificação		26	03/04/2022	16/04/2022	
		Implementação da análise e verificação		28	17/04/2022	30/04/2022	
		Testes de Validação		34	01/05/2022	11/06/2022	
Utilização & Suporte	TCC II	Coleta dos resultados	Captação da utilização em cenário real em projeto de pesquisa paralelo	35	12/06/2022	18/06/2022	
		Discussão dos resultados obtidos.		36	19/06/2022	25/06/2022	
Encerramento		Definição de próximas etapas	Estudo do caso de uso e sumarização dos resultados para apresentação da solução junto a banca	38	26/06/2022	09/07/2022	
		Formalização dos trabalhos e apresentação		39	10/07/2022	16/07/2022	

4 Análise e Discussão dos Resultados

4.1 Provisionamento de infraestrutura

4.1.1 Configuração Inicial

Foi utilizado para o setup inicial uma imagem customizada do Ubuntu *Server* 20.04, que se valia de uma configuração prévia automatizada por meio de [cloud-init](#). Isso possibilitou a execução de todas as máquinas mantendo a uniformidade de configurações iniciais de SSH, usuário e permissões de sudo em 2 horas, considerando que a execução foi a partir de uma única unidade de *pen-drive*. A melhor opção seria disponibilizar em uma unidade de rede comum as máquinas que pudessem ser acessadas durante o boot inicial.

Para assegurar um acesso seguro a rede do Winet foi selecionado um computador para servir como *load balancer* em rede privada e *bastion host* para acesso externo. Nesta configuração local foi realizada a instalação de um agente de tunelamento reverso de rede [ngrok](#) para expor a porta 22 em um endpoint com IP público, na qual uma comunicação SSH poderia ser estabelecida mediante a chave adequada. Essa configuração foi necessária para agilizar o desenvolvimento do provisionamento de segurança, enquanto um usuário de VPN não foi provisionado, e como a solução é semelhante a um tunelamento reverso de porta, em um endpoint efêmero, as configurações de rede da UFMG não foram expostas e pouco risco foi acrescido.

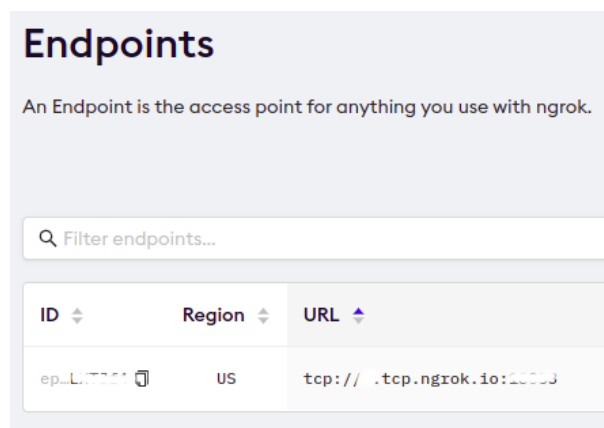


Figura 10 – Ngrok Endpoint

Juntamente a essa configuração foram realizadas as seguintes configurações de segurança básicas:

- instaladas fail2ban com throttle 3 requisições falhas por minuto, limitando ataques *brute-force* na porta SSH;

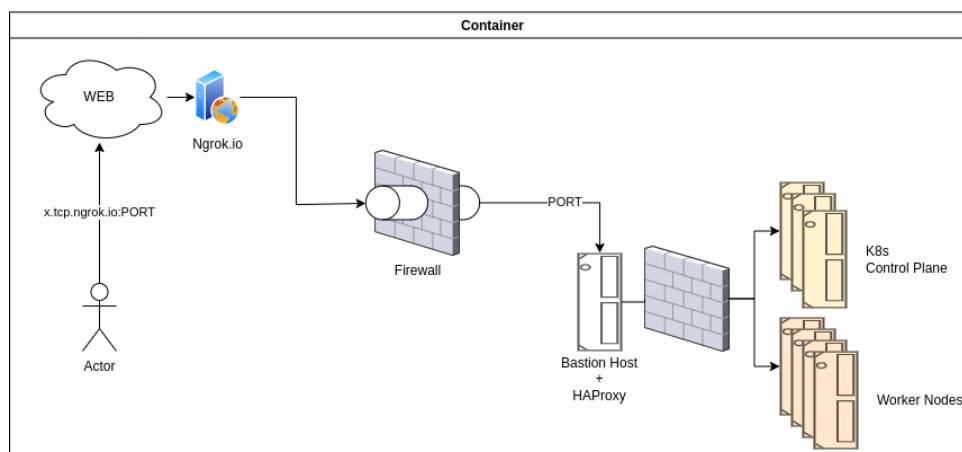


Figura 11 – Funcionamento do Ngrok

- foi desabilitado login root;
- foi desabilitada login por com senha;
- habilitado trafego na porta 22 de qualquer IP; e
- habilitado trafego em qualquer porta de IPs dentro da faixa de CIDR da subnet

4.1.2 Configurações dos computadores

Uma vez que uma conexão SSH foi estabelecida e o *firewall* todo *cluster* foi configurado utilizando o gerenciador de configurações (CMS) Ansible®. A configuração total do *cluster* demora entre 20-35 minutos sendo que a configuração foi realizada mais de uma vez do zero para garantir que todo o *playbook* (conjunto de configurações a serem executadas) fosse executado corretamente, considerando sua execução do início ao final e considerando a máquina sem nenhuma das configurações até seu estado de pronta. A execução idempotente (i.e operação realizada mais de uma vez sem que o resultado se altere) não foi alcançada para as configurações dos nós mestres devido à restrição de tempo e número de etapas a serem configuradas. Na figura 12 pode-se observar o diagrama de fluxo de execução lógica do *playbook*, porém essa não é a divisão modular do código, que é apresentado na forma de *roles*.

A formulação do inventário (que no Ansible® refere-se a relação de máquinas identificadas e/ou agrupadas, onde é possível especificar configurações e variáveis, para as mesmas) considera que os computadores a serem configurados estejam na mesma rede do bastion, sendo que toda a configuração passa por ele, garantindo assim a segurança da execução apenas para outros computadores, para os quais possa ser estabelecida conexão a partir dele. Esse inventário também divide em grupos as máquinas, podendo assim ser executado o *playbook* novamente para novas máquinas a serem adicionadas ao *cluster*, que garante o recrutamento estático de novos nós para comporem o *cluster*, e virtualmente podendo chegar no limite de rede no qual os nós do *cluster* estão agrupados. Para garantir

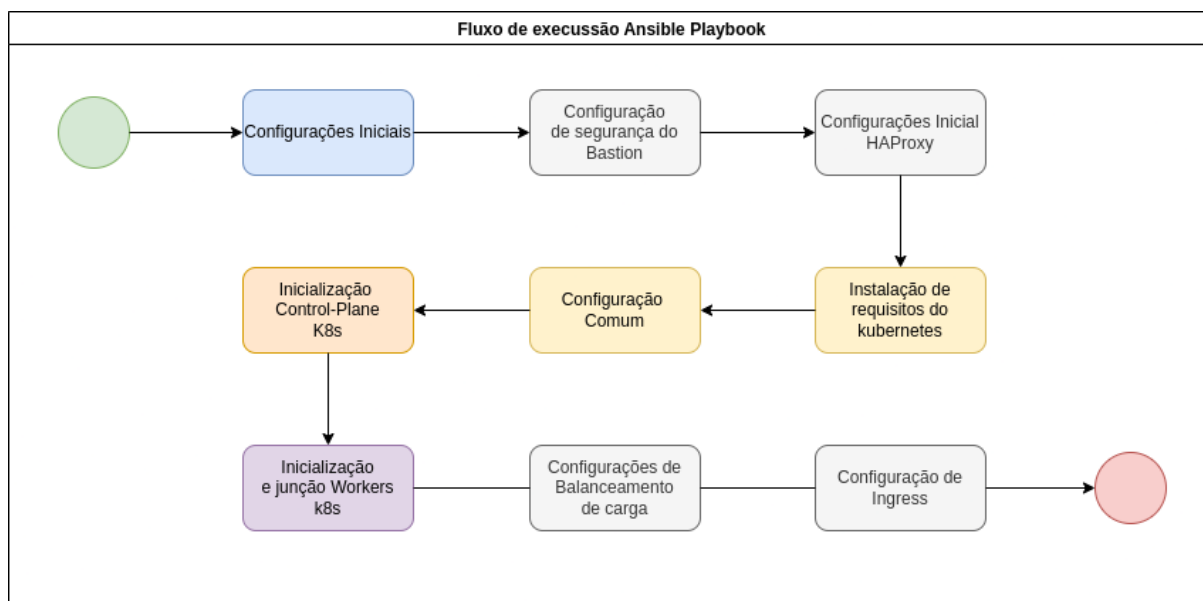


Figura 12 – Fluxo de execução do Ansible Playbook

a correta execução o inventário pode ser referenciado explicitamente durante a execução desde que tenha o mesmo formato do disponibilizado no repositório.

4.1.3 Configurações de infra do cluster

Duas ferramentas de vasto uso na configuração de infraestrutura e também do Kubernetes® foram utilizadas para garantir a gestão do provisionamento e configurações das aplicações que estão sendo executadas no *cluster*.

Terraform® é uma ferramenta de infraestrutura como código que permite definir recursos em arquivos de configuração legíveis. Essa ferramenta possibilita o provisionamento e gerenciamento de toda a infraestrutura por ela definida e seu ciclo de vida (provisionamento, alterações e decomissionamento). Essa ferramenta possui recursos que garantem a execução respeitando relações de dependência entre recursos, sejam elas diretas ou não.

Helm® é uma ferramenta de gerenciamento de pacotes para Kubernetes® tendo nela contida todas as configurações necessárias para configuração do *cluster* com uma dada aplicação.

O provisionamento de novas cargas de trabalho e *stacks* de processamento de dados foram realizados combinando o uso de Terraform®, que garantiu a correta gestão do estado dos recursos provisionados no *cluster*, com o Helm® que garantiu a correta configuração das aplicações configuradas no *cluster*. Essa combinação garantiu a máxima eficiência, gerenciando as dependências entre recursos e aplicações durante o tempo de execução, garantindo a imutabilidade dos recursos provisionados, o que diminui o *drift* ou

desvio de configurações, minimizando anti padrões como *snowflake*(MORRIS, 2020).

4.1.4 Gestão de armazenamento no *cluster*

Para gerenciar recursos que necessitam de persistência de dados neles contidos no Kubernetes® é necessário utilizar um conjunto de objetos de configuração específicos. Uma vez que container é um recurso efêmero, é necessário explicitar **o que** e **como** persistir dados que não devem ser perdidos quando o mesmo é desalocado no *cluster*.

Para isso existe um conjunto próprio de API (*application programming interface*) para persistir um conjunto de dados de um ou mais containers no Kubernetes®. *PersistentVolume* (PV) e *PersistentVolumeClaim* (pvc) são dois objetos de APIs que gerenciam persistência dinâmica de dados utilizando *StorageClass*. PV é um recurso que possui um ciclo de vida diferente e individual de um pod. PVC é uma solicitação de armazenamento dentro do *cluster*, que pode solicitar a um PV ou diretamente a um *StorageClass* (perfis de volumes). Sendo que no segundo, algumas configurações sobre políticas de acesso e qualidade de serviço podem ser previamente especificadas, deixando o requerente responsável apenas por configurações mais simples, como tamanho do volume necessário. Para configurações de volumes que utilizam protocolos específicos de sistema de arquivos, como NFS (*network file system*), pode se utilizar uma API específica chamada *container storage interface*(CSI) que abstrai especificidades de como um volume deve ser provisionado e como um container deve se anexar a ele.

StorageClass é a forma mais recomendada para gerenciamento de CSI para disponibilizar volumes compartilhados no *cluster* de Kubernetes®, porém em *clusters bare-metal* não existem muitas opções de *provider* CSI que não necessitem de um software ou ainda hardware específico, como no caso de SANs (*StorageAreaNetwork*) específicos. Sem CSI, as chamadas de *filesystem* que são executados pelos containers que alocam PVC (*persistent volumes claim*) podem incorrer em erros o que resulta na perda de dados ou ainda em erros de IO (*input output*).

Tendo como premissa que alguns volumes devem estar disponíveis para o cluster, e não ao container, não é possível utilizar PV (persistent volumes) ou *StorageClasses* que sejam locais. Ou seja, que não disponibilizem aquele volume ao *cluster* como um todo, uma vez que um pod ao ser encerrado por qualquer motivo pode ser alocado novamente em outro nó do cluster, que não possuirá qualquer registro da informação escrita anteriormente.

Com isso, resumimos a duas alternativas abertas: Ceph e NFS. Ambos os protocolos possuem CSI. Primeiramente, optou-se por utilizar o Ceph, uma vez que esse protocolo possibilitaria o uso de toda a capacidade de armazenamento do *cluster* de forma compartilhada, e com isso somaria-se os volumes disponíveis localmente, podendo resultar em um volume elástico que aumentaria toda vez que um nó fosse adicionado ao cluster, somando

a capacidade de armazenamento do *cluster* como um todo.

Após algumas tentativas de implementação, o funcionamento correto e esperado do driver não foi alcançado. E mesmo sendo a opção mais indicada, tendo um grau de complexidade que extrapolou o tempo disponível foi substituído pelo driver NFS, que garante a disponibilidade de volumes compartilhados, mas é baseado apenas no volume local do servidor que o provê. Mesmo tendo menos benefícios, ainda atende o propósito de volumes compartilhados, e pode ser usado em uma estratégia *mesh*, que garantiria uma soma de volumes entre os nós do cluster, mas que precisaria ser adaptado para ser disponibilizado como um pool de storageclasses, também fugindo do intento desse trabalho.

Por fim, visando a possível substituição do servidor NFS, por um NAS (*Network Area Storage*) que é uma solução de hardware específico ou genérico (configurável) e, portanto, sendo mais viável do ponto de vista econômico (preço/volume). Vale ressaltar que, o menor custo vem com problemas de latência, o que pode prejudicar o desempenho de soluções de processamento de dados que sejam intensivos em leitura e escrita de dados, especialmente o segundo.

Para tornar a solução viável em termos de complexidade de configuração, disponibilidade de hardware e custo, optou-se por usar o servidor bastion e *load balancer* como também o servidor de NFS. Como o número de requisições realizadas no *cluster* ainda não é tão grande, uma vez que a maior parte das requisições serão internas e de comandos, não foi mapeado como um risco de desempenho, mas tornando o uso mais intenso do cluster, seria recomendado isolar a função de *server* de NFS para um NAS ou servidor específico.

4.2 Disponibilização das imagens de container

Diversas dependências para a execução das tarefas e logo após a exploração visual e análise dos dados coletados pela orquestração devem ser incorporadas às imagens base utilizadas para o Airflow e também Jupyter. Para controlar a entrega dessas imagens e disponibilizá-las para o servidor, foi publicado em um repositório de imagens pública, com credenciais que expiravam em 12 horas. Essas imagens utilizaram uma estratégia tag para *release* associada ao *commit message* no repositório do git, podendo assim recuperar versões específicas e rastrear mudanças que possivelmente alteraram o comportamento esperado dos containers que as utilizarem.

Essa tag era então substituída por meio de variáveis no Terraform e assim utilizadas para configuração dos recursos durante o tempo de deploy das aplicações com as novas tags. Essa estratégia garante um fluxo auditável de mudanças e preconiza a imutabilidade, uma premissa para uso de infraestrutura como código.

4.3 Orquestração do processamento

Para orquestrar tarefas dentro do *cluster* garantindo a execução de atividades que tenham início, meio e fim, com alguma relação de dependência entre as atividades executadas, é necessário a utilização de uma aplicação capaz de orquestrar tarefas, i.e. gerenciar fluxos de trabalho. Nesse trabalho a ferramenta utilizada foi o Apache Airflow®. Essa ferramenta gerencia atividades utilizando um grafo direcional acíclico (DAG) semelhante ao Apache Spark®. Essas atividades ou tarefas serão aqui chamadas de *tasks*. Essas *tasks* e suas dependências são definidas em Python (uma linguagem de programação interpretada e de tipagem dinâmica) e o Airflow gerencia o agendamento (scheduler) e a execução (executor). A execução de uma DAG pode ser por meio de eventos ou agendamento (e.g. diariamente, por hora etc.).

A configuração do processo de ETL para ingestão dos dados do banco de medicamentos industrializados foi desenhado de acordo com o diagrama de fluxo da Figura 13 e implementado de acordo com a Figura 14.

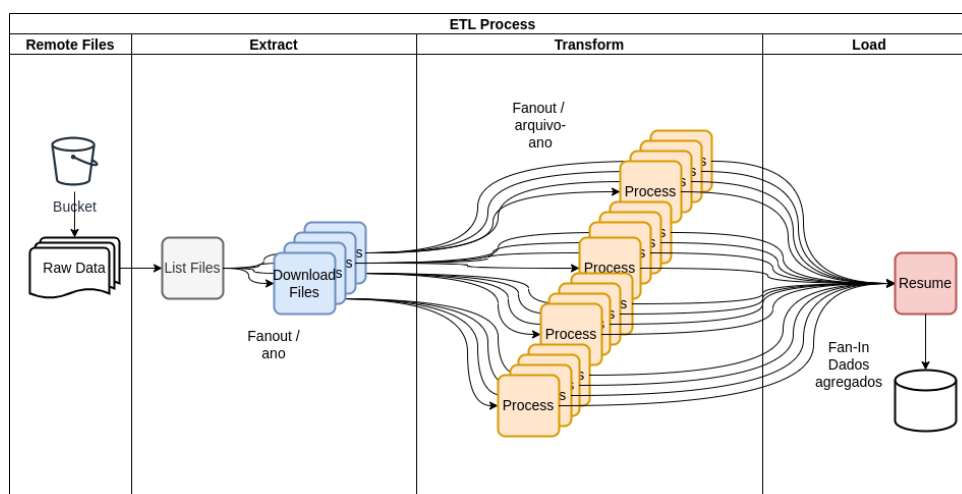


Figura 13 – Airflow - Diagrama de Fluxo

Para entender melhor a execução desse fluxo de trabalho é necessário entender como se dão as sequências de chamadas entre os componentes do Airflow. Na Figura 15 pode-se visualizar o diagrama de sequência onde cada life-line é um componente da aplicação, sendo que o scheduler, o trigger, o webserver são objetos de deployment dentro do kubernetes.

Objetos de deployment controlam objetos de replicaset que apenas controlam a quantidade de objetos pod que serão executados a partir de um modelo de configuração. Esse modelo, entre outras configurações, especifica memória, cpu, volumes, imagem de container etc. Tendo dito isso, também é necessário descrever o worker. Airflow *Worker* nada mais são que os componentes que de fato executam uma *task* especificada na DAG. Nesse trabalho, utilizamos o KubernetesExecutor para provisionar cada nova *task* executada

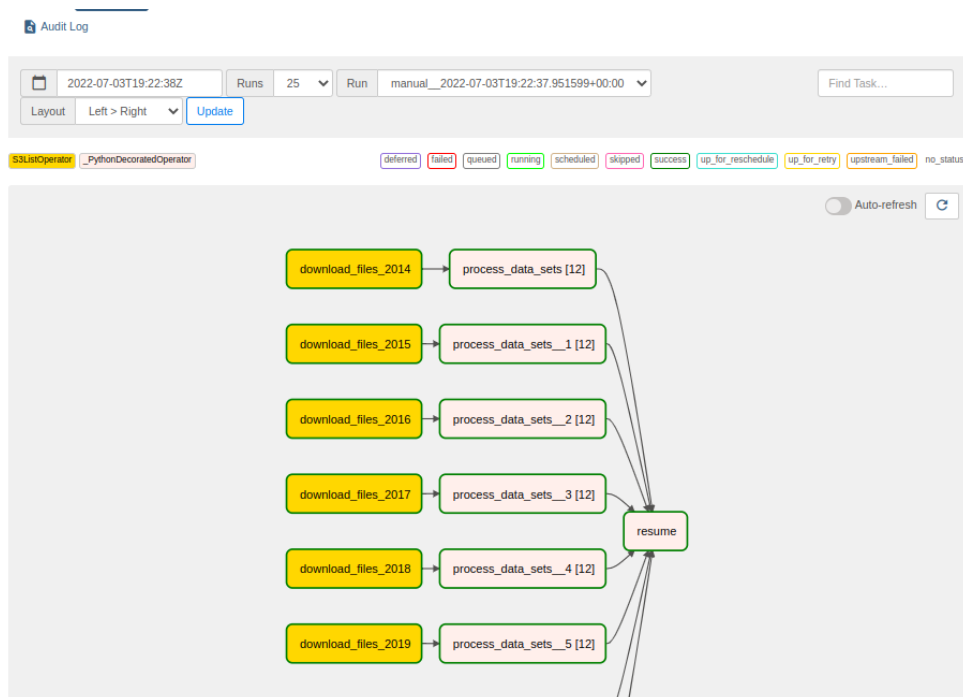


Figura 14 – ETL DAG

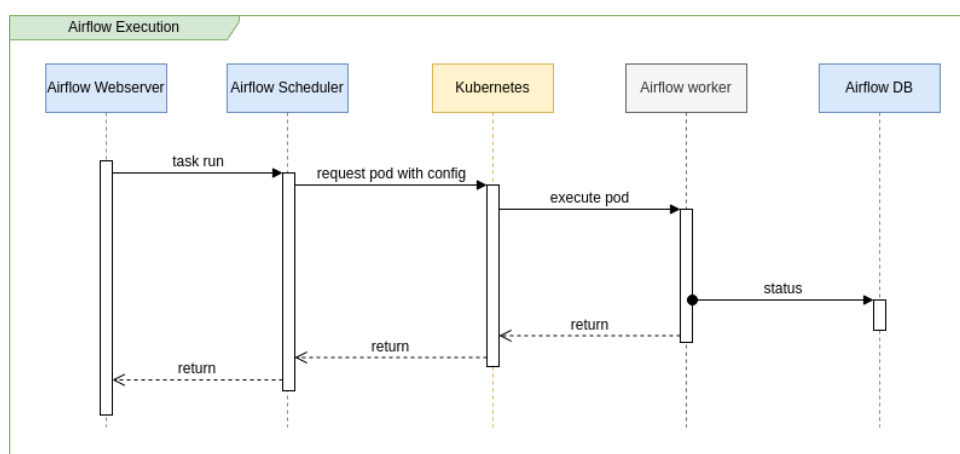


Figura 15 – Airflow - Diagrama de Sequência

no *cluster* em um novo pod, com recursos definidos na configuração da task, o que fica especificado em código python.

A configuração da *task* e como ela deve ser executada é passada a um módulo do Airflow *Scheduler* chamado executor. O Airflow possui diversos tipos de executors. Mais comumente utiliza-se o CeleryExecutor, porém para garantir que o *cluster* tenha alocação das pods dinamicamente optou-se pelo KubernetesExecutor. Esse componente nada mais é que um serviço que realiza chamadas de API para o provisionamento de pods que por sua vez executaram a task.

Apesar de não representada, há uma etapa anterior, um *operator* que lista os anos

disponíveis para criação de grupos de *mapper* de tarefas dinâmicas, é um *fan-out* de duas etapas, lançando os arquivos que irão listar anos disponíveis em bucket S3 na AWS, outro que listará os arquivos disponíveis de cada ano e seu download.

Na etapa de processamento seguinte, os dados são carregados desses arquivos em memória, em um *dataframe* do pandas (biblioteca python). Pelo tamanho dos arquivos, não é possível fazer o carregamento de uma só vez e por tanto a opção de chunks foi usada - isso particiona os dados para garantir o processo por partes. Essa opção garante que diversas tarefas sejam executadas em paralelo sem sobrecarregar o *cluster* ou mesmo causar eventos de OOM (*out of memory*), fazendo assim com que a tarefa entre na fila de *retry*.

O ponto de configuração dos containers foi de 1vCPU e 2GB de RAM para processar arquivos entre 0.5-0.7GB de dados por vez, como a tabela de escrita dos dados processado no PostgreSQL era a mesma para garantir a consolidação dos dados, o pipeline de execução de cada tarefa ficava entre 4 e 12 minutos desde o momento de início do *download* até o completo carregamento em banco. Como a operação de escrita em banco também é uma atividade restrita, ou seja, não permite duas escritas simultâneas, esse com certeza era um ponto de restrição do fluxo de dados. A possível solução para esse gargalo era a escrita dos dados pré-processados em tabelas distintas e em um passo seguinte a consolidação em tabela única. Porém, para manter a estrutura de um ETL simples e validar o processamento no cluster, não foi necessário realizar essas etapas de otimização.

Mencionadas as limitações já observadas, é necessário indicar mais uma importante restrição: o banco também está sendo executado no *cluster* com 1,5GB de RAM e 0.3 vCPU, sendo um recurso compartilhado entre diversos outros recursos do *cluster* por ter mais de um banco na mesma instância. E ainda utilizando CSI de NFS, que como mencionado anteriormente adiciona latência e conhecidamente não performa muito bem em usos intensos de escritas.








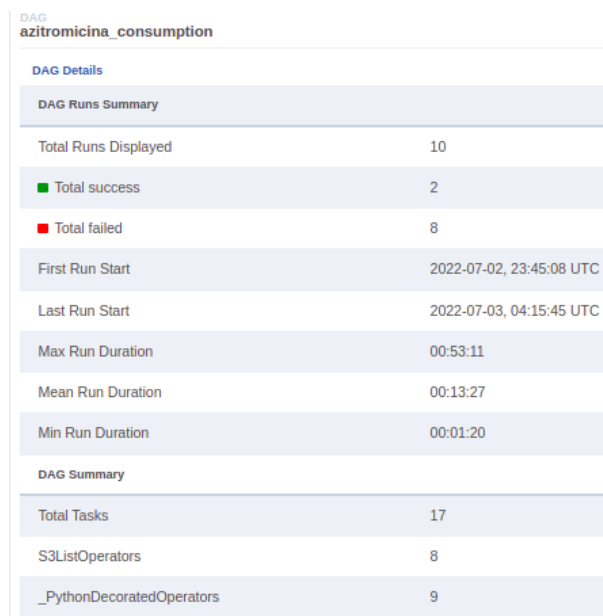
<input type="checkbox"/>	 EDA_Industrializados_201401.csv	653.0 MB
<input type="checkbox"/>	 EDA_Industrializados_201402.csv	623.3 MB
<input type="checkbox"/>	 EDA_Industrializados_201403.csv	666.2 MB
<input type="checkbox"/>	 EDA_Industrializados_201404.csv	693.3 MB
<input type="checkbox"/>	 EDA_Industrializados_201405.csv	737.0 MB
<input type="checkbox"/>	 EDA_Industrializados_201406.csv	701.4 MB
<input type="checkbox"/>	 EDA_Industrializados_201407.csv	720.6 MB

Figura 16 – S3 armazenamento dos dados

Considerando o tamanho e tempo médio de execução das atividades podemos inferir que:

- o tempo de execução total dessas tarefas de forma serial, seria de aproximadamente $8min * 90arquivos = 720min$ para importar todos os arquivos

- mesmo com as execuções simultâneas limitadas a 12, não utilizamos 50% do cluster, como demonstrado na Figura 19



The image shows a screenshot of the Airflow web interface displaying the DAG run summary for 'azitromicina_consumption'. The summary is divided into two sections: 'DAG Runs Summary' and 'DAG Summary'.

DAG Runs Summary	
Total Runs Displayed	10
■ Total success	2
■ Total failed	8
First Run Start	2022-07-02, 23:45:08 UTC
Last Run Start	2022-07-03, 04:15:45 UTC
Max Run Duration	00:53:11
Mean Run Duration	00:13:27
Min Run Duration	00:01:20

DAG Summary	
Total Tasks	17
S3ListOperators	8
_PythonDecoratedOperators	9

Figura 17 – Relatório de execução

É possível avaliar que a sobreposição de tarefas é gerenciada pelo Airflow mantendo a restrição de 12 execuções por vez, como demonstrado na Figura 18. Essa configuração foi limitada diretamente no Airflow durante o provisionamento, utilizando variáveis de ambiente. Essa restrição foi utilizada para restringir o número de escritas simultâneas que poderiam ocorrer no banco de dados que armazena a saída de cada uma das tarefas. Caso o número de concorrentes fosse máximo, duas coisas poderiam ocorrer: 1) o tempo para finalizar cada *task* poderia extrapolar o limite de tempo definido para cada *task* ser executada ou 2) o banco poderia ser sobrecarregado de requisições demorando ainda mais para inserir os dados em tabela, uma vez que teria que gerenciar as requisições de inserção simultâneas.

Porém, o tempo total gasto foi de 53 min, segundo a Figura 17. Isso representa 55% do tempo esperado, mesmo considerando as restrições de desempenho citadas anteriormente.

Pode-se inferir que o número de execuções simultâneas representavam 24vCPUs e 48GB de memória por limite de tarefas em execução, representando um servidor que sozinho custaria entre 20 e 40 mil reais, e nesse trabalho utilizou-se apenas máquinas que não estavam sendo utilizadas do DCC UFMG.

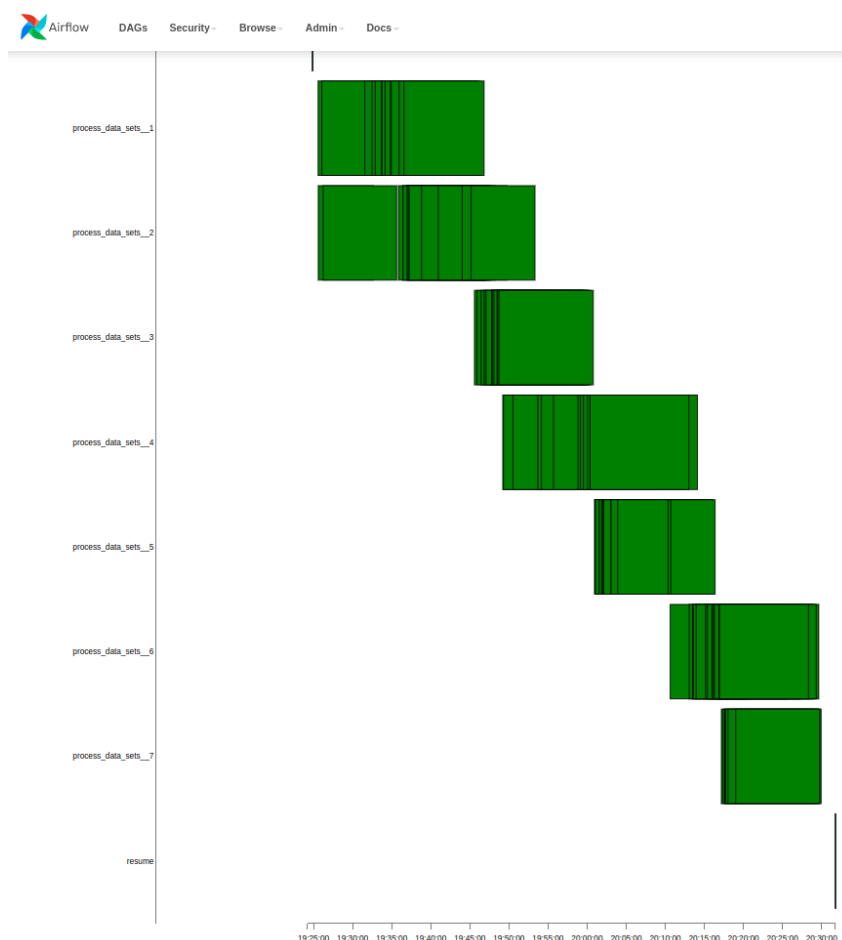


Figura 18 – Grafico de Gantt da orquestração

4.4 Desempenho do cluster

Devido as restrições colocadas para execuções simultâneas no cluster, garantimos que o *cluster* não fosse completamente utilizado, o que permitiria o seu uso para outras aplicações que viessem a ser executadas simultaneamente. Esse resultado atende a um dos objetivos específicos desse trabalho. Como mostrado na Figura 19 as atividades do *cluster* não ultrapassam em nenhum momento durante a execução do processo de importação e processamento dos dados. Essa avaliação utilizando o método USE de monitoramento permite analisar problemas de desempenho e identifica-los de maneira mais direta.

Para avaliar esse gráfico precisa-se identificar que as linhas amarela, azul e verde, representam os nós que pertencem ao *control-plane*, ou seja os nós que realizam as atividades de controle do *cluster* e parte de seu monitoramento. Esse monitoramento diz respeito apenas aos objetos que estão sendo supervisionados, ou seja, sob avaliação do control-manager e *scheduler* por meio do apiserver. As demais linhas são nós workers, que efetivamente executam as cargas de trabalho e visivelmente tem sua capacidade mais exigida durante o processo de importação e processamento dos dados.

Para a *task* de download, primeira etapa do fluxo de trabalho realizado no Airflow, é possível notar que o processo demanda mais escrita e leitura de disco e também possui maior atividade de rede e memória. Isso se deve ao fato de estar realizando tanto o download dos arquivos da fonte de dados, como também escrevendo o arquivo em disco. Após essa tarefa o uso de memória e disco se tornam cíclicos, essa ciclicidade se deve ao fato de que apenas 12 tarefas podem ser executadas por vez, como apresentado anteriormente nesse capítulo. Como todas as tarefas subsequentes escrevem o resultado em disco após a finalização pode-se observar que o resultado de escrita em disco e rede também é elevado, porém a diferença nessa etapa é que a escrita apesar de ser em disco, utiliza o volume de dados do container com CSI-NFS. Sendo um protocolo de arquivos de sistema em rede, esses arquivos são temporariamente escritos em disco e logo retransmitidos ao NFS *server* o que gera o tráfego de rede elevado no cluster, a semelhança do primeiro processo de download simples dos arquivos.

Para reforçar essa observação sobre a execução do processo, avalia-se o monitoramento dos nós realizado pelo control plane, que enxerga a capacidade total do cluster, ao invés de seus nós na figura 21. Já na figura 5 pode-se observar o comportamento sobreposto da utilização percentual de memória e cpu dos pods que executam as tarefas em relação ao cluster.

A diferença de métricas possibilita ter uma visão mais holística de como a aplicação esta utilizando o *cluster* em si, e também possibilita validar alguns pontos.

- O número de atividades em paralelo poderia ser aumentado, garantido assim melhor utilização do cluster, porém seria necessário alterar a lógica de processamento para escrever em tabelas individualizadas no banco de dados, e resumi-las na última etapa do processo.
- Protocolos de rede que garantissem o compartilhamento de dados entre os discos dos nós (como o Ceph) poderiam representar ganhos significativos, não sendo necessário trafegar todo o dado para fora do nó, quando houvesse uma escrita em disco.
- O processo de fanout utilizado

4.5 Análise de dados

Todas as análises foram executadas em um Jupyter notebooks provisionado no cluster.

Um total de 95.345.640 prescrições de azitromicina foram atendidas em farmácias e drogarias do Brasil, entre 2015 e 2021. A maioria dos pacientes para os quais foi dispensado o medicamento era do sexo feminino (53,62%), com média de idade de 32,75 ($\pm 2,04$). Entre as regiões com maior venda de azitromicina destacam-se as regiões Sudeste (47,44%) e Sul (22,47%) e entre as UF, destacam-se São Paulo (24,76%), Minas Gerais (13,17%) e Rio Grande do Sul (12,49%) Tabela 5

Características	n	%
	95345640	100
Sexo do paciente		
Feminino	50051932	53,62
Masculino	45293708	46,38
Região		
Centro Oeste	8325772	8,73
Nordeste	15311389	15,44
Norte	5050278	5,3
Sudeste	45232822	47,44
Sul	21425379	22,47
Unidade Federativa		
Acre	235293	0,25
Alagoas	587010	0,62
Amapá	251408	0,26
Amazonas	541218	0,57
Bahia	3672405	3,85
Ceara	2545617	2,67
Distrito Federal	1205841	1,26
Espírito Santos	1704011	1,79
Goiás	5007864	5,25
Maranhão	1414456	1,48
Mato Grosso	1068418	1,12
Mato Grosso do Sul	1043649	1,09
Minas Gerais	12552967	13,17
Pará	2640661	2,77
Paraíba	2207241	2,31
Paraná	5818312	6,1
Pernambuco	1823847	1,91
Piauí	861105	0,9
Rio de Janeiro	7372345	7,73
Rio Grande do Norte	1653179	1,73
Rio Grande do Sul	11911733	12,49
Rondônia	693489	0,73
Roraima	199669	0,21
Santa Catarina	3695334	3,88
São Paulo	23603499	24,76
Sergipe	546529	0,57
Tocantins	488540	0,51

Tabela 5 – Características das prescrições de azitromicina atendidas em farmácias e drogarias, Brasil, 2014-2020.



Figura 19 – Método USE monitoramento de hardware

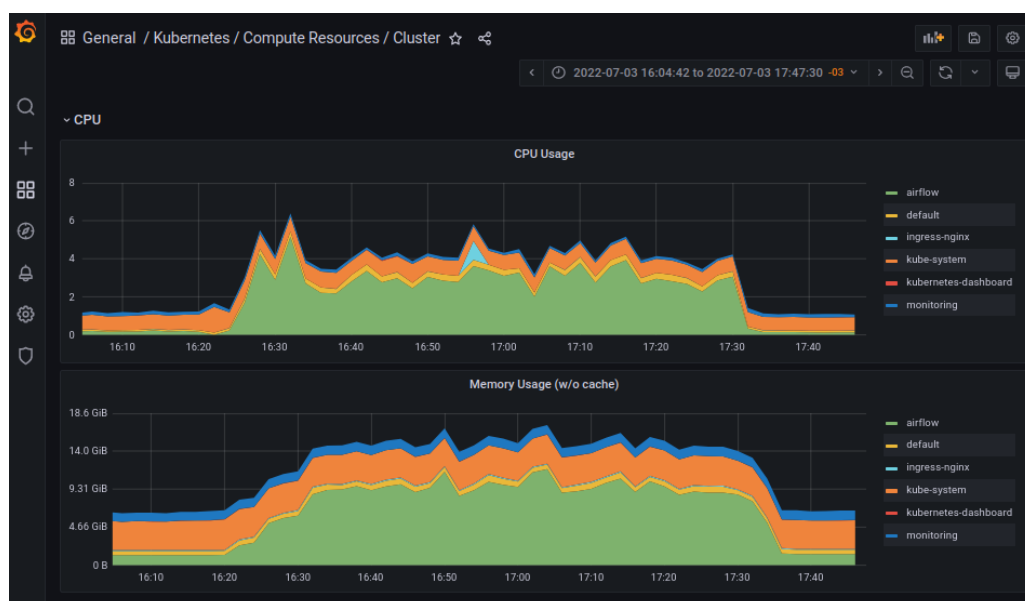


Figura 20 – Monitoramento do *cluster* pelo Control-Plane

No período de 6 anos, em números absolutos, o número de prescrições de azitromicina, no Brasil, aumentou de 13.421.249 prescrições em 2014 para 17.735.901 prescrições

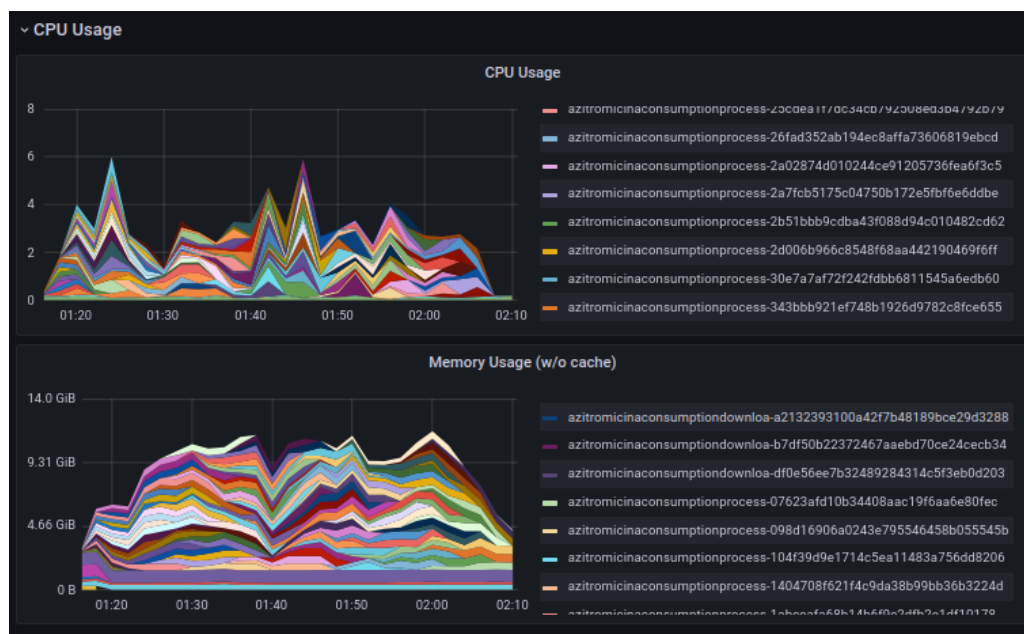


Figura 21 – Perfil da aplicação - Utilização dos pods de processamento em relação ao cluster

em 2020, representando um aumento de 32,1%. Considerando o crescimento da população, isso representa um aumento na taxa de prescrição de azitromicina de 66,2 para 83,8 prescrições por 1.000 habitantes, em todo o período Tabela 6.

A maioria das UF apresentaram aumento nas taxas de prescrição de azitromicina por 1.000 habitantes, com destaque para: Minas Gerais (66,2 para 149,2 prescrições por 1.000 habitantes), Rondônia (37,5 para 109,4 prescrições por 1.000 habitantes) e Roraima (37,2 para 99,8 prescrições por 1.000 habitantes). Considerando apenas os anos de 2019 e 2020, como períodos pré e durante a pandemia da COVID-19, respectivamente, observa-se que houve aumento nas taxas de prescrição no Brasil e em todas UF, exceto no Rio de Janeiro (87,3 para 76,2 prescrições por 1.000 habitantes) e no Rio Grande do Sul (106,6 para 97,4 prescrições por 1.000 habitantes) (Tabela 7).

Os maiores aumentos nas taxas de prescrição de azitromicina foram observados em Minas Gerais (81,7 para 149,2 prescrições por 1.000 habitantes), Paraná (66,3 para 129,2 prescrições por 1.000 habitantes), Roraima (38,9 para 99,8 prescrições por 1.000 habitantes), Rondônia (58,8 para 109,4 prescrições por 1.000 habitantes) e Goiás (74,3 para 116,1 prescrições por 1.000 habitantes).

Pode se observar os numero absolutos na Figura 23

Pode-se observar que o resultado é coerente quando avaliado sob a perspectiva do Tau de Kendal como descrito na Figura 24, que avalia a correlação de crescimento entre variáveis. Os estados do Amazonas, Mato Grosso, Mato Grosso do Sul, Paraná, Pará, Rio Grande do Norte, Santa Catarina e Tocantins, apresentam um alto grau de

Localidade	Taxa prescrição / 10 ⁴ hab	
	2014	2020
Brasil	66,2	83,8
Unidade Federativa		
Acre	30,2	71,9
Alagoas	20,5	42,8
Amapá	36,5	79,2
Amazonas	13	38
Bahia	46,3	51,5
Ceara	23,8	38,3
Distrito Federal	52,7	96,9
Espírito Santos	49,6	97,8
Goiás	133,2	116,1
Maranhão	19,6	31,1
Mato Grosso	31,3	73,5
Mato Grosso do Sul	49,4	79,2
Minas Gerais	66,2	149,2
Pará	30,6	49,4
Paraíba	145,2	101,7
Paraná	57,8	129,2
Pernambuco	26,2	39,3
Piauí	116,8	33,7
Rio de Janeiro	44,1	76,2
Rio Grande do Norte	50	106,7
Rio Grande do Sul	137,6	97,4
Rondônia	37,5	109,4
Roraima	37,3	99,8
Santa Catarina	61,3	84,5
São Paulo	96,6	86,7
Sergipe	30,7	58,6
Tocantins	34,5	80,6

Tabela 6 – Taxas de prescrição de azitromicina atendidas em farmácias e drogarias, no Brasil, em 2014 e 2020

correlação de crescimento no tempo e com significância estatística (valor $p < 0.05$). Apesar dos resultados para o estado de Minas Gerais descrito acima, o mesmo não apresenta significância estatística, o que pode ser causado pelo aumento populacional do estado, o que pode mascarar o aumento da taxa de prescrição.

Localidade	Taxa prescrição / 10 ⁴ hab	
	2019	2020
Brasil	59,9	83,8
Unidade Federativa		
Acre	32,3	71,9
Alagoas	23,5	42,8
Amapá	45,9	79,2
Amazonas	17,3	38
Bahia	31,9	51,5
Ceara	29,7	38,3
Distrito Federal	57,3	96,9
Espírito Santos	62,5	97,8
Goiás	74,3	116,1
Maranhão	18,5	31,1
Mato Grosso	49,2	73,5
Mato Grosso do Sul	56,4	79,2
Minas Gerais	81,7	149,2
Pará	27,3	49,4
Paraíba	62,3	101,7
Paraná	66,3	129,2
Pernambuco	27,2	39,3
Piauí	22,1	33,7
Rio de Janeiro	87,3	76,2
Rio Grande do Norte	73	106,7
Rio Grande do Sul	106,6	97,4
Rondônia	58,8	109,4
Roraima	38,9	99,8
Santa Catarina	73,3	84,5
São Paulo	68,1	86,7
Sergipe	36	58,6
Tocantins	44,1	80,6

Tabela 7 – Taxas de prescrição de azitromicina atendidas em farmácias e drogarias, no Brasil, em 2019 e 2020

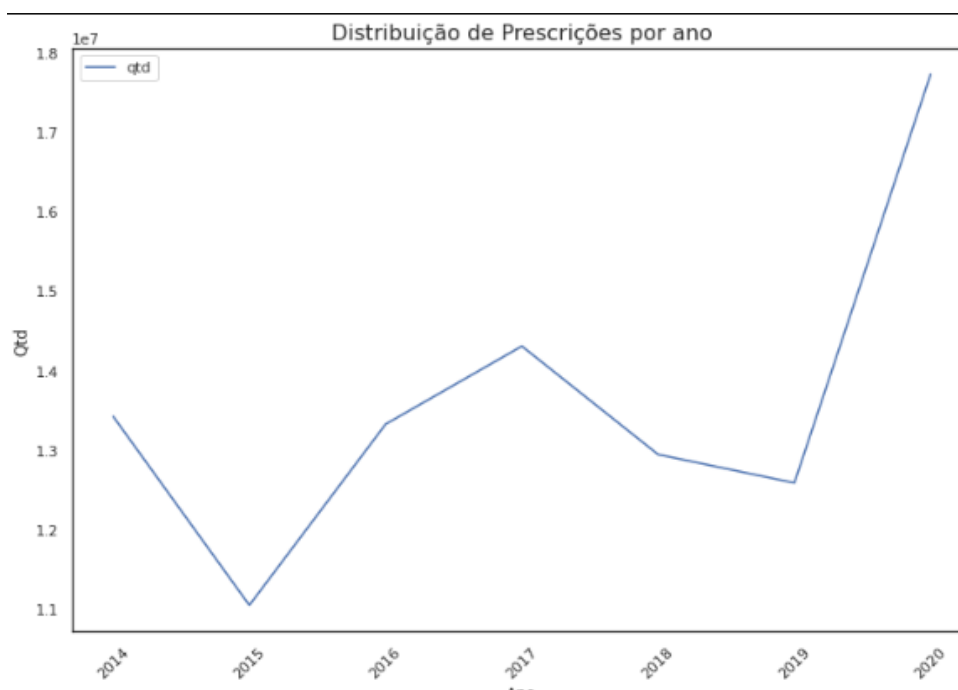


Figura 22 – Prescrição por ano

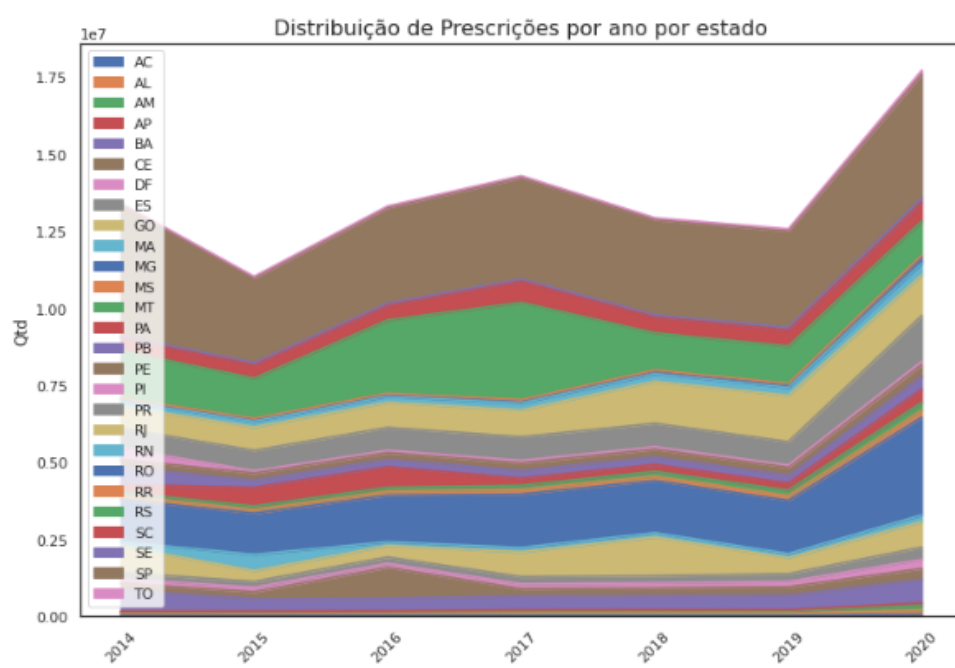


Figura 23 – Prescrição por ano por estado

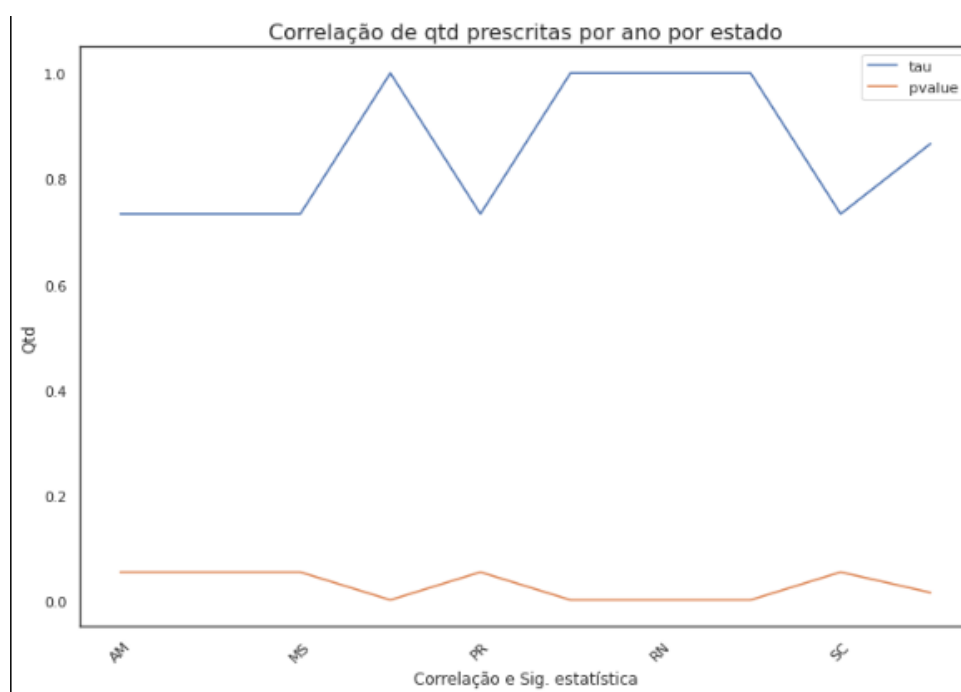


Figura 24 – Avaliação de correlação de crescimento por ano por estado utilizando tau de Kendall

5 Conclusão

Diversas tecnologias foram incorporadas e avaliadas para a orquestração das tarefas de processamento e importação de dados a serem utilizadas e *cluster* Kubernetes®, o que se apresentou viável como alternativa econômica, para análise de grandes volumes de dados utilizando computadores comuns e reaproveitados. A eficiência energética não foi um tema abordado no trabalho, por entender-se que as máquinas reaproveitadas já estariam ligadas em laboratórios de informática da universidade, e por vezes subutilizadas. Outro caso que esse reaproveitamento pode amenizar, diz respeito ao ciclo de vida dessas máquinas no inventário de patrimônio em instituições públicas. A integralização desses equipamentos ao *cluster* possibilita sua utilização mais eficiente (aproveitamento de máquinas ociosas) e prolonga a utilidade desses equipamentos (utilidade até descarte). Mesmo quando esses equipamentos estão defasados, ao integrar o *cluster* podem receber cargas de trabalho mais propícias às suas configurações.

O provisionamento do *cluster* apresentou um tempo de configuração adequado para o projeto, não representando um percentual muito extenso do tempo disponível para realização desse trabalho. Sendo a maior parte do tempo gasto para finalização das configurações, relativos ao estudo das ferramentas utilizadas e sua correta configuração e integração.

Sobre a avaliação de desempenho, o tempo de execução da importação e processamento dos dados foi adequado ao volume de dados apresentado, comparando cargas de trabalho semelhantes na experiência do autor. Porém, foram identificados diversos pontos de melhoria e otimizações possíveis. Essa identificação deixa claro que a integração de profissionais de múltiplas especialidades é essencial para utilizar essa alternativa como factível em estudos maiores e mais exigentes. Porém, essa alternativa se mostra promissora, especialmente sob o ponto de utilização de recursos subutilizados, com o processo de recrutamento através de configuração dinâmica de gerenciadores de configurações.

Sobre o consumo de azitromicina, supõe-se que o estímulo ao seu consumo como opção de tratamento para a COVID-19, dentro do ‘kit COVID’, possa ter influenciado o aumento das prescrições e, conseqüentemente, o aumento do consumo observado em 2020. O que indica que ações de saúde podem apresentar viés político e influenciar decisões em saúde individuais entre a população.

É importante que novas estratégias de avaliação de dados em saúde sejam propostas e validadas, especialmente na produção e viabilização do uso de ferramentas que contextualizem o cenário de investimento em pesquisa e educação do nosso país, de modo a tornar viável produzir soluções e pesquisas cada vez mais rapidamente no campo

da saúde, ajudando a comunidade técnica e não técnica a tomarem melhores decisões embasadas em informações produzidas pela comunidade científica.

Restrições orçamentárias não só impactam na velocidade, mas também na viabilização de projetos científicos que se valem de tecnologias. Na análise e processamento de dados públicos, devido ao volume de dados produzidos no Brasil, isso se torna um empecilho e projetos como esse se tornam ainda mais necessários.

O desenvolvimento do presente trabalho possibilitou a avaliação da aplicação de orquestração de cargas de trabalho de análise de dados de prescrições de azitromicina, advindo de uma base consideravelmente grande. Por meio da revisão bibliográfica foram identificadas outras plataformas de orquestração de cargas de trabalhos e foi possível fazer uma avaliação crítica baseada nos requisitos disponibilizados nas documentações oficiais, bem como a avaliação crítica do propósito ao qual o presente trabalho se propunha. Ainda na literatura, foi possível encontrar dados e informações a respeito dos impactos socioeconômicos resultantes da restrição orçamentária a pesquisa de uma forma geral. O que torna possível a tomada de decisão em saúde pautada em dados e também auxilia a população a ter melhor noção, baseada em dados, da situação de saúde onde se encontra e assim poder auditar os órgãos públicos responsáveis pela condução do SUS e políticas de saúde associadas.

5.1 Trabalhos Futuros

O estudo de estratégias de otimização do dimensionamento de recursos e avaliação comparativa de mais tecnologias podem tornar ainda mais plausível a utilização de computadores ‘desktops’ subutilizados nas universidades para provisionamento de ambientes de análise de grande massas de dados. Permitindo assim propor estratégias e ferramental indicados como solução para essas análises, bem como uma estratégia viável de utilização de um conjunto de computadores quando ociosos. Como trabalhos futuros aponta-se a avaliação de algoritmos que capturem as métricas das execuções e avaliem a melhor configuração de recursos para cada task, visando otimizar tempo de execução e um objetivo de taxa de utilização do cluster. Outro trabalho possível de ser realizado é a pré-configuração de clusters em todos os laboratórios de informática e bibliotecas das universidades públicas, sendo estes federados a um *cluster* mestre, que mediante a restrições de horário poderia distribuir tarefas aos seus federados, utilizando todos os computadores das universidades em caso de necessidade. Outro trabalho proposto seria um painel para submissão de atividades de análises para processamento em um *cluster* central das universidades por outros departamentos, programas de pesquisas e instituições utilizando um conceito de cloud privada, porém em kubernetes semelhante à ideia proposta pela plataforma OpenShift da RedHat.

Referências

AIRFLOW, A. **Architecture Overview**. 2022. Disponível em: <<https://airflow.apache.org/docs/apache-airflow/stable/concepts/overview.html>>. Citado na página 18.

ANDRADE, A. Q. d. A tomada de decisão e sistemas de informação em saúde. jan. 2008. Accepted: 2019-08-13T14:49:40Z Publisher: Universidade Federal de Minas Gerais. Disponível em: <<https://repositorio.ufmg.br/handle/1843/ECIC-7XMFGC>>. Citado na página 7.

BRASIL. **DECRETO Nº 8.777, DE 11 DE MAIO DE 2016**. 2016. <http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2016/decreto/d8777.htm>. Citado na página 1.

BRITO, G.; TERRA, R.; VALENTE, M. T. Monorepos: A Multivocal Literature Review. **arXiv:1810.09477 [cs]**, out. 2018. ArXiv: 1810.09477. Disponível em: <<http://arxiv.org/abs/1810.09477>>. Citado na página 18.

CLOUDSTACK. **Installation Guide — Apache CloudStack 4.16.0.0 documentation**. 2022. Disponível em: <<https://docs.cloudstack.apache.org/en/latest/installguide/index.html>>. Citado na página 13.

COMER, D. **The Cloud Computing Book: The Future of Computing Explained**. [S.l.]: CRC Press, 2021. Google-Books-ID: QCs0EAAAQBAJ. ISBN 978-1-00-038427-7. Citado na página 16.

CORPORATION, I. **Big Data Analytics: Intel's IT Manager Survey on How Organizations Are Using Big Data**. [S.l.], 2012. Citado na página 7.

DIJCKS, J.-P. **Oracle: Big data for the enterprise**. [S.l.], 2013. Citado na página 7.

ETCD. **Install**. 2022. Section: docs. Disponível em: <<https://etcd.io/docs/v3.5/install/>>. Citado na página 17.

FACELI, K. et al. Inteligência artificial: uma abordagem de aprendizado de máquina. 2011. Citado na página 7.

GALVÃO, A. B.; VALENTIM, R. A. d. M. Desafios para os Avanços da Análise de Big Data na Saúde. In: **Anais Estendidos do Simpósio Brasileiro de Computação Aplicada à Saúde (SBCAS)**. SBC, 2019. p. 155–160. ISSN: 2763-8987. Disponível em: <https://sol.sbc.org.br/index.php/sbcas_estendido/article/view/6301>. Citado na página 1.

GNU. **A Quick Guide to GPLv3 - GNU Project - Free Software Foundation**. 2022. Disponível em: <<https://www.gnu.org/licenses/quick-guide-gplv3.en.html>>. Citado na página 16.

GREGG, B. **The USE Method**. 2022. Disponível em: <<https://www.brendangregg.com/usemethod.html>>. Citado na página 19.

- HUGE, M. **Different Types of Benchmarks**. 2022. Disponível em: <<https://www.cs.umd.edu/users/meesh/cmsc411/website/projects/morebenchmarks/types.html>>. Citado na página 19.
- KUBERNETES. **Kubernetes Documentation | Kubernetes**. 2022. Disponível em: <<https://kubernetes.io/docs/home/>>. Citado na página 17.
- LANEY, D. et al. 3d data management: Controlling data volume, velocity and variety. **META group research note**, Stanford, v. 6, n. 70, p. 1, 2001. Citado na página 7.
- MACCARTY, S. **A Practical Introduction to Container Terminology**. 2018. Disponível em: <<https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction#>>. Citado na página 9.
- MEHTA, N.; PANDIT, A. Concurrence of big data analytics and healthcare: A systematic review. **International Journal of Medical Informatics**, v. 114, p. 57–65, jun. 2018. ISSN 1386-5056. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1386505618302466>>. Citado na página 1.
- MEHTA, N.; PANDIT, A. Concurrence of big data analytics and healthcare: A systematic review. **International Journal of Medical Informatics**, 2018. Citado na página 7.
- MELL, P.; GRANCE, T. **The NIST Definition of Cloud Computing**. [S.l.], 2011. Disponível em: <<https://csrc.nist.gov/publications/detail/sp/800-145/final>>. Citado na página 13.
- MORRIS, K. **The Snowflakes as Code antipattern**. 2020. Disponível em: <<https://infrastructure-as-code.com/book/2021/11/19/snowflakes-as-code.html>>. Citado na página 25.
- OPENSTACK. **OpenStack Docs: Xena Installation Guides**. 2022. Disponível em: <<https://docs.openstack.org/xena/install/>>. Citado na página 13.
- PLATFORM definition and meaning | Collins English Dictionary. 2022. Disponível em: <<https://www.collinsdictionary.com/dictionary/english/platform>>. Citado na página 13.
- PORTNOY, M. **Virtualization essentials**. [S.l.]: John Wiley & Sons, 2012. v. 19. Citado na página 9.
- RESENDE, L.; VIANA, L.; VIDGAL, P. **PROTOCOLOS CLÍNICOS DOS EXAMES LABORATORIAIS**. 1. ed. Minas Gerais: Secretaria de Estado de Saúde de Minas Gerais, 2009. Citado na página 7.
- RFC4254. 2006. Disponível em: <<https://datatracker.ietf.org/doc/html/rfc4254>>. Citado na página 18.
- SAHOO, J.; MOHAPATRA, S.; LATH, R. Virtualization: A survey on concepts, taxonomy and associated security issues. In: **2010 Second International Conference on Computer and Network Technology**. [S.l.: s.n.], 2010. p. 222–226. Citado na página 8.
- SANTOS-PINTO, C. D. B.; MIRANDA, E. S.; CASTRO, C. G. S. Osorio-de. O “kit-covid” e o programa farmácia popular do brasil. **Cadernos de Saúde Pública**, SciELO Brasil, v. 37, 2021. Citado na página 20.

SCHEEPERS, M. J. Virtualization and containerization of application infrastructure: A comparison. In: **21st twente student conference on IT**. [S.l.: s.n.], 2014. v. 21. Citado na página 19.

SCHROECK, M. et al. Analytics: el uso de big data en el mundo real. **IBM Institute for Business Value, Oxford, Informe ejecutivo**, 2012. Citado na página 7.

TRUYEN, E. et al. A Comprehensive Feature Comparison Study of Open-Source Container Orchestration Frameworks. **arXiv:2002.02806 [cs]**, mar. 2021. ArXiv: 2002.02806. Disponível em: <<http://arxiv.org/abs/2002.02806>>. Citado 2 vezes nas páginas 13 e 14.

VERMA, A. et al. Large-scal *cluster* anagement at Google with Borg. In: **Proceedings of the European Conference on Computer Systems (EuroSys)**. Bordeaux, France: [s.n.], 2015. Citado na página 14.