SEMANA ACADÊMICA
REVISTA CIENTÍFICA
ISSN 2236-6717

Aplicação do Sistema *Docker* a fim de Eliminar Desperdício no Setor de Desenvolvimento de *Software*

Jeison Pereira de Oliveira [1]
https://orcid.org/0009-0002-8651-1882
Marcelo Albuquerque de Oliveira [2]
https://orcid.org/0000-0003-2496-646X

RESUMO

O progressivo emprego de sistemas computacionais, bem como a procura por criação e manutenção de *software* também crescem, continuamente, dificultando as ações das pequenas equipes centralizadas. O desenvolvimento de *software* tem se desenvolvido em sintonia com aparecimento de novas tecnologias e as demandas do mercado, sempre mais globalizado, obrigando um grande número de empresas a subdividirem suas equipes de desenvolvimento de *software* a fim de ampliar sua capacidade operacional. Este estudo tem por objetivo a implementação do Sistema Operacional *Docker Host*, visando a erradicação de desperdícios no setor de desenvolvimento de *software*. No que diz respeito, especificamente, ao *Docker host* ele auxilia os desenvolvedores de *softwares,* gerando uma acessibilidade de dados mais ampla, seguida de uma independência de dados capaz de proporcionar um aumento na rapidez da tramitação dos processos. Em razão desses aspectos a presente pesquisa se caracteriza como bibliográfica, a partir de uma abordagem qualitativa, exploratória e explicativa. Conclui-se com a análise de que este estudo favorecerá a melhoria no desempenho do desenvolvedor de *software*, procurando reduzir o tempo que, anteriormente, era utilizado para configurar ambientes e desenvolver *softwares*, tornando mais célere a tramitação dos processos, dentro das organizações, com benefícios para todo o conjunto da sociedade.

Palavras-chave:
Software; Docker Host; Desenvolvimento.

[1] Mestrando do Curso de Engenharia de Produção. Bacharel em Sistemas de Informação. Universidade Federal do Amazonas, Amazonas. E-mail: jeison.po09@gmail.com
[2] Professor Orientador. Pós-Doutor em Empreendedorismo e Inovação pela Universidade Fernando Pessoa em Portugal. Professor da Universidade Federal do Amazonas. E-mail: marceloolivera@ufam.edu.br

http://dx.doi.org/10.35265/2236-6717-235-12648
FORTALEZA-CE. EDIÇÃO 235. V.11. ANO 2023.

Application of The Docker System in Order to Eliminate Waste in The Software Development Sector

ABSTRACT

The progressive use of computational systems, as well as the demand for software creation and maintenance also grow continuously, making the actions of small centralized teams difficult. Software development has developed in line with the emergence of new technologies and the demands of the increasingly globalized market, forcing a large number of companies to subdivide their software development teams in order to expand their operational capacity. This study aims to implement the Docker Host Operating System, aiming to eradicate waste in the software development sector. With regard, specifically, to the Docker host, it helps software developers, generating broader data accessibility, followed by data independence capable of providing an increase in the speed of the processing of processes. Due to these aspects, this research is characterized as bibliographical, based on a qualitative, exploratory and explanatory approach. It concludes with the analysis that this study will favor the improvement in the performance of the software developer, seeking to reduce the time that, previously, was used to configure environments and develop software, making the processing of processes faster, within organizations, with benefits for society as a whole.

Keywords:

Software; Docker Host; Development.

Submetido em: 13/06/2023 – **Aprovado em**: 11/07/2023 – **Publicado em**: 14/07/2023

1 INTRODUÇÃO

Um número significativo de setores de Tecnologia da Informação e Comunicação (TIC), tanto em órgãos estatais como em empresas privadas têm por hábito estabelecer moderações ou controles baseados em procedimentos operacionais desenvolvidos por elas mesmas, dando origem a praxes ou protocolos orientados a silos tecnológicos destinados a proteção de informações geradas pelas empresas, capazes de impedir vazamentos e utilização inadequada de dados.

Como resultado dessa ação Caruso e Steffen (1999), indicam que cresceu o nível de complexidade dos ambientes de tecnologia, tornando obrigatória a formação de equipes de administração cada vez maiores e com maior *expertise*, sobretudo, no tocante a infraestrutura. Nessa conjuntura, o desenvolvimento de respostas ou soluções integradas por causa das necessidades de negócio, constitui-se em um grande obstáculo a ser superado, principalmente devido aos embaraços para uniformizar plataformas que, geralmente, foram estruturas de forma estanque ou separada para cada sistema.

De forma complementar, uma contradição que põe em risco a eficiência das instituições, ao mesmo tempo em que os técnicos ou especialistas de infraestrutura empregam todo o empenho para manutenção da administração, equilíbrio e consistência do ambiente, aspectos que potencialmente diminuem a rapidez para o provimento de versões mais recentes de software, técnicos de desenvolvimento são chamados a oferecer aplicabilidades inovadoras para suas necessidades, de forma cada vez mais imediata. Tais circunstâncias reiteram a necessidade de que se reservem instrumentos computacionais de forma mais eficiente e versátil, aumentando a importância de condutas com o poder de operar em ambientes de infraestrutura de forma mais produtiva e proveitosa (CACIATO, 2009).

Na expectativa de suprir essa carência, o modelo de computação em nuvem (*cloud computing*), cada vez mais adotado na atualidade, vem se constituindo em alternativa eficiente, não somente para a indústria de TIC, mas também para a TIC corporativa nos mais diversos campos da economia. Este padrão ou referência se resume na apresentação de recursos computacionais em forma de serviço, cuja infraestrutura e os *softwares* implicados são potencial e dinamicamente instanciados com dedicação mínima de gerenciamento, apoiados em parcerias firmadas entre o provedor e o cliente (BOTACIM *et al*., 2016).

Diversas são as teorias tratando das facilidades tecnológicas da atualidade e sobre a forma como elas têm concorrido para o aumento da produtividade, uma vez que aceleram procedimentos e tornam mais simples o acesso de dados, levando um grande número de organizações públicas a, cada vez mais, elegerem um ambiente digital.

O emprego das ferramentas tecnológicas nos setores públicos gerou consequências positivas, tornando mais eficiente a procura por meios capazes de transformar, positivamente, os padrões operacionais relacionados aos serviços prestados, alterando os discordantes pontos de vista sobre o trabalho, além de uma mudança nos processos empregados pelos colaboradores assegurando evolução sob todos os aspectos (GONÇALVES, 2021).

É muito fácil identificar os inúmeros obstáculos existentes no dia a dia administrativo

e operacional das instituições públicas e, em especial, no tocante as atividades relativas à área de desenvolvimento de *softwares*, para onde são carreados um sem-número de projetos de manutenção e produção de sistemas.

No intuito de resolver essa questão a utilização do *Docker* se propõe a oferecer um ambiente normatizado e ajustável para a criação de *softwares* com unificação de configurações, por meio de dois simples arquivos (*Dockerfile* e/ou *Docker-compose.yml*) capazes de permitir a geração das imagens e *containers* desde que o equipamento no qual se pretende criar ou executar o sistema seja dotado de uma versão do cliente *Docker*.

Assim, o estudo em questão tem por objetivo a implementação do Sistema Operacional *Docker Host*, para erradicação do desperdício no setor de desenvolvimento de *software*.

Falando, especialmente, sobre o *Docker host* é possível compreender que esta tecnologia apoia os desenvolvedores de *softwares* promovendo uma acessibilidade de dados mais ampla, seguida de autonomia de dados, aumentando a rapidez com que os processos se desenvolvem. A partir desses aspectos é possível compreender esta pesquisa como sendo de caráter bibliográfico, através de uma abordagem qualitativa, exploratória e explicativa (SCHAEFER; NARDI, 2020).

A partir de todos esses elementos, estabelece-se como expectativa o desenvolvimento de uma pesquisa ou exame sobre a forma através da qual o presente artigo irá favorecer o aprimoramento da performance de cada desenvolvedor de *software* na busca pela atenuação, radical, do tempo anteriormente empregado na estruturação de ambientes e desenvolvimento de *softwares*, acelerando os processos internos das organizações públicas, gerando benefícios, também para a sociedade como um todo.

2 REFERENCIAL TEÓRICO

2.1 Docker

A partir de seu desenvolvimento, em 2013, o *Docker* se popularizou como saída ou recurso para criação, consecução e administração de *containers*. Tal popularidade do *Docker* se deve ao fato de que ele, além de descomplicar o uso dos *containers*, também iniciou ou instituiu o seu fornecimento por meio da nuvem (VITALINO; CASTRO, 2018).

O *Docker* é um *software* de código aberto para emprego no desenvolvimento, organização e operacionalização de aplicações em ambientes com sistema operacional GNU/Linux. Por meio do aparelhamento da plataforma em questão no sistema operacional originário, existe a possibilidade de que sejam criados contêineres que protegem as aplicações em ambientes isolados por completo. Os procedimentos necessários ao desenvolvimento de um *contêiner*, nessa circunstância, englobam o relacionamento do usuário com a plataforma *Docker*, situada no sistema operacional originário. Essa plataforma, no que lhe toca, dialoga com o *kernel* do sistema para a composição dos *namespaces* que servirão de isolamento para

o *container* (DOCKER, 2017).

O *Docker* é uma aplicação própria para o desenvolvimento e administração de *containers,* altamente popular no mercado. Ele possibilita o desenvolvimento e a administração de ambientes isolados, permitindo o acondicionamento de uma aplicação ou ambiente, dentro de um *container*, assumindo portabilidade para qualquer outro equipamento que permita sua instalação (GUEDES, 2018).

O *Docker* segue o padrão da virtualização sem que na verdade o seja, na prática. A partir do atributo da utilização de *containers* passa a ser desnecessário pôr em funcionamento todo um sistema operacional virtualizado, com todos os seus vínculos e seu *kernel* próprio, sobre o sistema operacional da máquina real para que se possa colocar em funcionamento a aplicação pretendida (SILVA, 2017).

O *Docker* será operacionalizado sobre o sistema operacional hospedado na máquina real, separado da mesma, ainda que dividindo o mesmo *kernel*, razão pela qual é compreendido como um novo processo. E é justamente em função de sua particularidade operacional, como um processo separado dentro do sistema hospedeiro, o *Docker* despreza o carregamento de um novo sistema operacional por inteiro, assumindo um novo *kernel* e seus vínculos ou dependências, tal como se daria em outras modalidades de virtualização (OLIVEIRA *et al*., 2019).

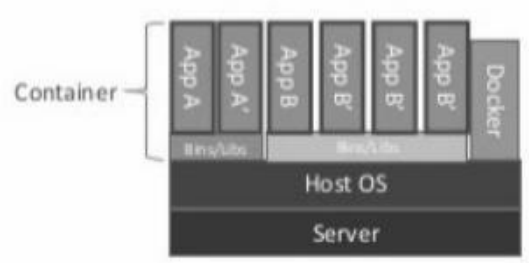
A partir da ideia ou concepção do *Linux Containers,* na perspectiva de que, somente, o programa necessário do sistema convidado seja carregado sobre o sistema hospedeiro, o *Docker* gera impacto reduzido na *performance* da máquina física, acelerando a operação pela capacidade de partilhar os mesmos recursos de hardware com segurança e constância (MAIA; MARTINS, 2020).

Um dos aspectos que fazem do *Docker* uma aplicação diferenciada é o fato dele possuir uma *engine* para o *deploy* de aplicações, rodando em um ambiente virtualizado de *container*. Criado para ser um ambiente desembaraçado e ágil no qual é possível rodar aplicações, além de *workflows* para o *deploy* dessas mesmas aplicações (TURNBULL, 2014).

Habitualmente, um *container* Docker é inicializado em menos de um segundo, já que não é exigido um *hypervisor,* nem é preciso ativar um sistema operacional *guest,* o que amplia o desempenho e a capacidade de suportar aumento elevado de carga sem perda de desempenho do ambiente. Para iniciar um *container* necessita-se, apenas, que uma máquina rodando um sistema operacional se harmonize com o *Linux* e o *Docker* (TURNBULL, 2014).

O propósito maior da utilização do *Docker* é permitir a operacionalização de ambientes isolados, no interior de uma mesma máquina, ainda que de forma aberta ao mundo externo. Na sequência apresenta-se a Figura 1, na qual é exibido um ambiente *Docker* (GOMES; SOUZA, 2015).

Figura 1. Ambiente Docker.



Fonte: Gomes e Souza (2015).

No processo de criação de programas permite a composição de ambientes análogos, tanto para o desenvolvimento, como para os testes e para a produção, de forma resumida utilizando *containers Docker*, propiciando o desenvolvimento e manutenção de aplicações e de seus respectivos ambientes.

2.1.1 Componentes do Docker

O *Docker* é integrado por inúmeros seguimentos que se comunicam para a criação da plataforma. Evidenciam-se, entre tais seguimentos, o Clienteservidor *Docker*, as *Docker Images* os *Registries* e os *Containers*. Estando em conjunto, os referidos seguimentos constituem o *core* do *Docker* e são fundamentais para o uso da plataforma. Para concluir a forma ou estrutura, também é admissível ter uma *API REST* para que se realize o diálogo entre Cliente *Docker* com o servidor ou *daemom*.

A) Cliente-Servidor *Docker*

*Docke*r utiliza uma forma ou natureza cliente-servidor e no caso específico do *Docker client* a comunicação é processada com o *Docker server* ou *daemon*. De maneira sucinta o cliente dirige comandos para o servidor, que é o elemento responsável pela execução de todo o trabalho pesado. É viável que se opere o *Docker client* e o *Docker server* tanto na mesma máquina como em máquinas distintas, permitindo que o cliente dialogue com um servidor remoto. O diáglogo entre o *client* e o *daemon* se processa por meio de *sockets* ou por meio de uma *API REST*.

B) Docker Images

Docker Images são a base de todo e qualquer tipo ou padrão de *container*, bem como respondem por sua execução. São, também, os responsáveis pela construção dos *containers*. Neste arquivo de imagem (*Docker Image*) ficam insertas as normas ou rotinas que determinam de que forma o *container* será estruturado. É possível reconhecer tais normas ou rotinas como o código fonte do *container*, uma vez que determinam a forma através da qual deverá ser constituído e organizado. Existe a possibilidade de se repartir, guardar e posteriormente atualizar *Docker Images*, que também são potencialmente portáveis (TURNBULL, 2014).

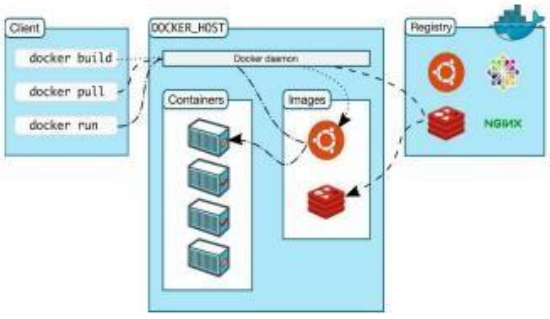
C) Registries

Pode ser caracterizado como um programa servidor que possibilita o aprovisionamento e compartilhamento das *Docker Images* e são classificados em duas modalidades distintas, a saber: públicos e privados. A modalidade pública do referido aplicativo recebe o nome de *Docker Hub*, sendo mantido por uma companhia denominada *Docker Inc.* É permitido a qualquer indivíduo abrir uma conta no *Docker Hub* com o objetivo de ratear e aprovisionar as imagens que produz. O *Docker Hub* armazena um número superior a 10 mil imagens geradas por indivíduos do grupo ou coletivo, em todo o planeta, totalmente, à disposição, para utilização. O *Docker Hub* também dispõe de meios para oportunizar o armazenamento de imagens em módulo privado, no qual se pode aprovisionar, juntamente com outros arquivos de caráter informativo-proprietário, em total segurança e potencialmente capazes de compartilhamento, exclusivo, para determinados componentes. Além desses aspectos existe a alternativa de desenvolver um novo *Registry* privado, uma vez que a codificação empregada para realização dos registries é livre e pode operacionalizado por qualquer indivíduo (TURNBULL, 2014).

D) Containers

Docker favorece ou auxilia a composição e operacionalização de *containers*, a partir dos quais se pode rodar aplicativos e serviços. Os *containers* são ativados com base em imagens, sendo capazes de comportar um determinado número de processos. *Containers Docker* são, na essência, representativos de um ambiente de execução que englobam um grupamento de procedimentos básicos. Todo *container* é munido de um *software* capaz de processar um composto de procedimentos ou comandos tais como iniciar, criar, reiniciar, entre outros. O *Docker* não leva em consideração o objeto em execução em cada um dos *containers*, isto é, independentemente de que no interior do *container* esteja sendo operacionalizado um servidor *web* ou uma base de dados, o carregamento de cada um dos *containers* é processado de forma idêntica. O programa ou aplicação também não leva em conta o local no qual o *container* está sendo operacionalizado. Como forma de exemplificar tal processo deve-se observar a seguinte possibilidade: um *container* é iniciado em um *laptop*, armazenado em um *registry* e o mesmo *container* ser iniciado em um servidor ou uma máquina virtual. *Containers* foram feitos para serem leves, portáveis e o mais genéricos o quanto possível.

Figura 2. Arquitetura do *Docker*.



Fonte: Docker Inc. (2016).

Como é possível observar na Figura 2, anteriormente exibida, é possível identificar uma arquitetura completa do *Docker*, mostrando o cliente, o servidor, o *Docker Images*, *Containers* e *registries*.

2.2 Virtualização

A virtualização é um mecanismo ou recurso apoiado em um programa que se utiliza do repartimento do hardware (máquina física) para apresentar o desempenho de múltiplos conceitos ou conteúdos de máquinas virtuais autônomas e separadas umas das outras (REDHAT, 2019).

A virtualização vem sendo utilizada desde a década de 1965 pela IBM em seus *mainframes* e tem se caracterizado como um processo bastante poderoso, sobretudo, em relação ao aumento dos meios ou recursos computacionais. Tal tecnologia é acessível desde 2001 na plataforma x86, o que significa dizer que os computadores padrão da indústria conseguem acessá-la (CACIATO, 2009).

De maneira geral, a virtualização opera na perspectiva de potencializar a utilização do hardware e reduzir, o quanto possível, os períodos de repouso do hardware. Dessa maneira se pode utilizar o potencial pleno de uma máquina física, dividindo em frações menores os recursos para diferentes setores ou áreas como, por exemplo, a virtualização de programas, de computadores convencionais, de servidores de aplicação, de atividades de armazenamento, de tarefas e funções de rede. A partir da virtualização, as corporações inclinam-se no sentido de aplicar um menor aporte de recursos na aquisição de ambientes computacionais completos, por causa das destinações dos recursos computacionais serem mais bem elaboradas e acessíveis (VMWARE, 2019).

A virtualização consiste num processo de emulação de uma máquina, através da utilização de um programa (*software*), rodando no interior de uma máquina real (física). Isso significa que uma determinada aplicação cria, virtualmente, dentro de uma máquina física, uma área específica e totalmente isolada, com todos os recursos de hardware, para operação em separado. Esse recurso permite que um determinado computador passe a ser capaz de rodar, simultaneamente, dois sistemas operacionais distintos, com padrão de memória RAM, tempo de processamento e execução de tarefas, também, individualizados (CACIATO, 2009).

Reis (2017) explica que a virtualização é, nos dias de hoje, um processo quase que unanimemente utilizado em ambientes de serviços de rede, sobretudo por se tratar de um recurso tecnológico que pode oferecer a condição de uso de apenas uma configuração física (hardware), compartilhada entre dois equipamentos virtuais expandindo, acentuadamente, as perspectivas para que se possa utilizar recursos relacionados a inúmeros serviços como servidores web, DNS, e-mail, além de firewall.

Andrade (2018) vê, na virtualização, o caminho para que se alcance o desenvolvimento da computação em nuvem. Corporações como a *Amazon,* por exemplo, empregam, de forma maciça, essa ideia com o objetivo de oferecer um grande número de serviços tais como a hospedagem de serviços web, servidores de arquivos e outros mais.

Serviços em nuvem são cada dia mais habituais e a partir da multiplicação dos smartphones, um número cada vez maior de serviços, que até então rodavam, somente, em *Desktops* ou *Notebooks*, têm sido transformados em aplicativos próprios para execução virtual para acesso através daquele tipo de aparelho telefônico. A partir disso os denominados data centers, dotados de configuração envolvendo *hardware* e *software* adequados para assumirem todo o comprometimento em relação ao processamento, com disponibilidade e capacidade variável de adaptação de suas proporções, além de credibilidade e sigilo, em formato específico para satisfazer tais requisitos (REIS, 2017).

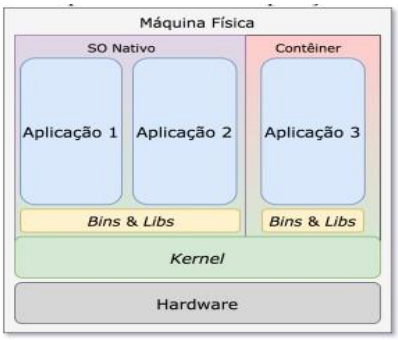
No meio corporativo, a virtualização é potencialmente capaz de oferecer inúmeros vantagens, uma vez que hoje as empresas vêm deslocando um expressivo montante de seus serviços para *Data Centers*, por várias razões diferentes, como a segurança, a redução dos custos com a permanência de servidores. Apesar disso, colaboradores locais continuam sendo encarregados da operacionalização de atividades como serviços de impressão, DHCP, *firewall*, autenticação, além de outras propósitos referentes aos negócios da empresa (CARISSIMI, 2008).

Realmente, ao se tratar de ambiente computacional, principalmente no tocante a serviços interconectados em rede, passa a ser fundamental falar sobre a tecnologia de virtualização.

2.3 Contêineres

Para Fulay (2017), Contêineres se constituem em métodos ou processos de virtualização, no que diz respeito a sistema operacional para ambientes *Linux*. Esse recurso tem por base a apartação de recursos característicos do *kernel* no sentido de que aplicações sejam executadas como se estivessem sendo realizadas em um ambiente isolado. Uma representação elementar do funcionamento de contêiners pode ser vista na Figura 3. Com destaque para o kernel do sistema operacional que é dividida entre aplicações e contêineres. Apesar disso, a maneira através da qual é possível estabelecer o contato com o kernel (sob a perspectiva das aplicações) não é igual, uma vez que o conjunto de aplicações (*Bins*) e bibliotecas (*Libs*) do sistema não é obrigatoriamente igual ao disponibilizado pelo sistema operacional original (FULAY, 2017).

Figura 3. Exemplo de isolamento de aplicações em contêineres.



Fonte: Adaptado de Docker (2017).

Tanenbaum (2003), em relação a execução, indica que há três instrumentos básicos em um *kernel Linux* que admitem o desenvolvimento e o uso de *containers*, a saber: *chroot, cgroups* e *namespaces*.

- *Chroot* (**Change Root**) deve ser entendido como a primeira iniciativa no sentido de promover o isolamento de processos no mundo *UNIX*. Adotado em 1979, o *chroot* é um comando que serve para rodar uma aplicação, ou *Shell,* determinando um diretório *root* diferente do *standard*. Assim, a aplicação executada com *chroot* não poderá conectar um outro diretório fora da árvore de diretórios determinada pelo comando, gerando a ideia de que o processo em questão segue executando em um ambiente isolado (TANENBAUM, 2003).
- *Cgroups* (*Control Groups*) é um recurso do *kernel* usado para administrar processos do sistema como o tempo de processador, memória e recursos de rede. Por meio de *cgroups*, administradores de sistema conseguem controlar e consultar o uso de recursos que são compartilhados entre processos de usuários. Essa função é essencial para possibilitar que sejam definidos limites de uso dos recursos em *containers*, sempre que for preciso (TANENBAUM, 2003).
- *Namespaces* são conteúdos que guardam ou revestem recursos do sistema, produzindo camadas de isolamento entre processos. Assim, processos próprios do *namespace* de um recurso têm a ciência de que possuem um âmbito ou domínio isolada do mesmo. Alterações nesse recurso são disponíveis, somente, aos processos que compartilham o mesmo *namespace,* ficando ocultos para outros processos (TANENBAUM, 2003).

Os instrumentos anteriormente mencionados já bastam para que sejam estruturados ambientes virtualizados, dentro de um sistema operacional GNU/Linux. Apesar disso, a atribuição de gerenciar recursos do *kernel,* por meio de tais instrumentos, manualmente, é bem complicada, exigindo bastante *know-how* a respeito da forma como funciona o sistema operacional. Essa dificuldade reduziu o nível de utilização daquele recurso, até bem pouco tempo. A disseminação de c*ontainers* ocorreu a partir da criação de instrumentos que descomplicam a API concedida pelo *kernel Linux*, simplificando a geração de ambientes isolados, para o usuário final (FULAY, 2017).

A referida descomplicação se estabelece, principalmente, por causa da criação de um espaço de execução em imagens de *containers*. Por imagem, nessa circunstância, entenda-se a estruturação de um espaço virtual em uma configuração passível de salvamento em determinado tipo de aprovisionamento, para que esse mesmo ambiente possa ser reproduzido múltiplas vezes. Um contêiner significa, nesse caso uma imagem que está sendo processada. O vínculo existente entre imagem e *container* equivale à associação existente entre arquivo executável e processo: ao mesmo tempo em que a imagem diz respeito uma reprodução em disco, ao *container* é cabe cuidar execução dessa reprodução no processador (GRATTAFIORI, 2016).

Igualmente, no tocante as imagens de *container*, é fundamental ressaltar que a concepção do ambiente de execução não gera um sistema operacional finalizado. Uma imagem de contêiner possui bibliotecas próprias e programas de sistema, mas não abrange o *kernel Linux* que é provido pelo sistema operacional original, sendo compartido entre os

sistemas originais e demais *container*s em atividade (FULAY, 2017).

2.4 Lean Office

O *Lean Office* se apoia nos fundamentos do *Lean Manufacturing* ajustado para as atribuições administrativas, significando dizer, lançar mão das práticas da produção magra e não do raciocínio linear ou contínuo da linha de montagem (LAGO; CARVALHO; RIBEIRO, 2008).

Dessa forma, *Lean Office* não tem relação com corte de pessoal e recursos, mas sim, com o empenho das pessoas gerando valor para o cliente, extinguindo desperdícios, agilizando a operação, abreviando o tempo de inoperância originado pela papelada e burocracia.

Tapping e Shuker (2003, *apud* LOPES, 2011, p.12) manifestam a ideia de que "muitas organizações procuram "fazer o *Lean*" sem, necessariamente, tornarem-se *Lean*" uma vez que uma empresa que busca assumir uma administração preocupada com a mitigação e eliminação de gastos desnecessários ou esbanjamento, deve assumir uma nova forma de pensar e agir, reavaliar sua cultura, olhar de forma diferente os desperdícios, independentemente da área ou atividade em que eles ocorram.

Roos, Sartori e Paladini (2011) salientam que a ideia de utilizar o raciocínio e as condutas da indústria para os procedimentos do setor de serviços, constituem-se em experiência das mais ousadas para estudiosos sobre a área de serviços e profissionais do ramo. É bastante comum que a adoção do *Lean office*, por parte das empresas, gere algumas dificuldades iniciais, justamente pelo fato de que grande parte das rotinas se referem a informações e conhecimentos, enquanto a ferramenta em questão trata de informações e serviços. Tal inadequação acaba por tornar bastante complicada a identificação de esbanjamentos ou dilapidações. A configuração do *Lean Office* apresenta prognósticos de conjunturas de atuação quase imperceptíveis, uma vez que procedimentos tratando de fluxos não materiais ou corpóreos (ROOS; SARTORI; PALADINI, 2011).

Os padrões de desempenho dos trabalhadores e dos clientes são, efetivamente, sugestionados pelos atributos dos departamentos administrativos, motivo pelo qual torna-se muito importante recombinar as atividades de cada área ou divisão, abrindo espaço para que colaborem, decisivamente, para geração de bens e riquezas e tratar das carências específicas de trabalhadores ao longo de todo o processo, transmitindo aos mesmos uma verdadeira sensação de que, de fato, fazem farte de tudo que envolve a obtenção de resultados (CARDOSO, 2013).

3 CONCLUSÃO

Computação em nuvem é, hoje, um modelo ou padrão crucial para se assegurar o acesso a aplicações e serviços. Uma das bases para utilização desse modelo é a virtualização. Cada formato de virtualização é dotado de orientações, projetos e objetivos distintos, impedindo que se distinga qual é superior aos demais no mercado.

O *Docker* é uma das opções de virtualização e se constitui na essência da pesquisa desse estudo. O *Docker* é dotado de um descomplicado processo de instalação. O ambiente Linux é o que possui maior compatibilidade com essa aplicação, uma vez que se utilizado um outro ambiente, será exigido o emprego de uma máquina virtual para adquirir um *kernel Linux* e então executar o *Docker* o que consequentemente gerará que, rotineiramente, gera um déficit de performance, por causa da camada suplementar da máquina virtual.

Independentemente de que a experiência apresentada seja descomplicada, ficou claro o poder do *Docker*, uma vez que ele é dotado de potencial para gerar crescimento e agregar valor aos negócios das instituições, além de atender as necessidades dos clientes sem pressionar, de forma significativa, os custos e facilitando o gerenciamento dos *containers*. Na eventual necessidade de um *container* de um servidor web, com poucos comandos, se poderia utilizar este mesmo servidor operando e acessível para ser utilizado.

Em relação as ações de desenvolvimento de aplicações, o *Docker* também pode ser a opção, considerada a simplicidade e rapidez com que se poderia ativar todo o ambiente. Nesta hipótese não haveria a necessidade de que o desenvolvedor permanecesse, por horas, preparando ou se capacitando para configurar inúmeros *softwares* e ambientes, bastando, apenas, criar uma imagem ou um *Dockerfile* para emprego em todos os terminais de desenvolvimento.

Outra alternativa seria a utilização da tecnologia para a reprodução de ambientes, considerando que se todos os servidores utilizarem a mesma imagem ou *Dockerfile* para desenvolvimento dos *containers,* todos irão dispor do mesmo ambiente de execução, tendo como única forma de diferenciação o seu potencial físico.

Resumidamente, o emprego de *containers* abre espaço para que sejam incorporados vários benefícios próprios de virtualização a ambientes de computação paralela. Levando em consideração os benefícios analisados, é possível apontar o isolamento de aplicações, aprimoramento da reprodutibilidade baseada em ambiente de execução uniforme, troca ou mudança (portabilidade) de ambientes de execução e a condição de que o usuário estabeleça a distribuição Linux e quais versões de bibliotecas serão empregadas no ambiente. Em relação ao último elemento é, particularmente, curioso considerando-se que, normalmente, administradores de um *cluster* de computação paralela não oferecem regalias administrativas para os usuários, que, como resultado, não são capazes de mudar, com facilidade, as configurações do sistema operacional original ou nativo (como bibliotecas padrão) da máquina.

Ainda que o uso preliminar do *Docker* seja realizado de forma facilitada, é preciso relativa expertise para a aplicação de certos conceitos. A documentação fornecida pelo próprio site do *Docker* é ampla e de fácil compreensão. A comunidade *Docker* também tem grande atuação na *internet*, com inúmeros fóruns e tutoriais que ajudam no aprendizado.

REFERÊNCIAS

ANDRADE, J. **O que é um hypervisor?** 2018. Disponível em: http://www.getcard.com.br/novo/o-que-e-um-hypervisor/ Acesso em: 16 mar. 2023.

BOTACIM, R. S.; ATHAYDE, S. S.; OLIVEIRA, F. M.; XAVIER, B. M.; SOUZA, M. Computação nas nuvens: evolução e peculiaridade dos serviços e da segurança da informação. **Revista Interdisciplinar do Pensamento Científico**, [*s.l.*], v.2, n.1, p. 259-277, 2016.

CACIATO, L.E. **Virtualização e consolidação dos servidores do datacenter.** Centro de Computação da Universidade Estadual de Campinas– São Paulo. Campinas, 2009. Disponível em: https://www.ccuec.unicamp.br/cicau/download/Artigo_Virtualizacao_Datacent er.pdf Acesso em: 16 mar. 2023.

CARDOSO, G.O.A. Análise crítica da implementação do lean office: um estudo de casos múltiplos. **Gestão da Produção, Operações e Sistemas**, Bauru, v.8, n.1, p.23-35, Jan. 2013.

CARISSIMI, A. Virtualização: da teoria a soluções**.** In: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 26, 2008. **Anais** {...]. Porto Alegre: UFRS, 2008. p. 174 – 206.

CARUSO, A. A.C.; STEFFEN, F.D. **Segurança em informática e de informações**. 2. ed. São Paulo: SENAC, 1999. DOCKER DOCS. **Docker documentation**. 2016. Disponível em: https://docs.docker.com/ Acesso em: 16 mar. 2023.

DOCKER. 2017. Disponível em: https:///www.docker.com Acesso em: 14 mar. 2023.

FULAY, A. **Containers Deep Dive – LXC vs Docker**. San Jose: Robin Systems Inc., 2017.

GOMES, R.; SOUZA, R. **Docker**: infraestrutura como código, com autonomia e replicabilidade. Salvador: Universidade Federal da Bahia, 2015. Disponível em: http://www.ixwticifes.ufba.br/modulos/submissao/Upload-275/66257.pdf Acesso em: 16 mar. 2023.

GONÇALVES, R. L. G. **Proposição de melhorias aos processos de uma biblioteca**: uma aplicação do Lean Office. 2021. 44 f. Monografia (Iniciação Científica do curso de Engenharia de Produção apresentado a Área de Ciências Exatas e Aplicadas) - Centro Universitário Sagrado Coração. Bauru, 2021.

GRATTAFIORI, A. **Understanding and hardening Linux Containers. Version 1.**1. [S.I]: NCC Group, 2016.

GREEF, A.C.; FREITAS, M. do C.D.; ROMANEL, F.B. **Lean office**: operação, gerenciamento e tecnologias. São Paulo: Atlas, 2012.

GUEDES, M. **No final das contas**: o que é o Docker e como ele funciona? 2018. Disponível em: https://www.treinaweb.com.br/blog/no-final-das-contaso-que-e-o-docker-e-como-ele-funciona Acesso em: 14 mar. 2023.

LAGO, N.; CARVALHO, D.; RIBEIRO, L.M. Lean office. **Fundição**, Portugal, v.2, p.248-249, 2008.

LOPES, M.C. **Melhoria de processo sob a ótica do lean office.** 2011. 68f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Produção Mecânica) – Universidade de São Paulo, São Carlos, 2011.

MAIA, H. M.; MARTINS, P. J. **Análise de uso da tecnologia de software Docker aplicando Conteinerizacão na computação em nuvem**. 2020. 12 f. Trabalho de Conclusão de Curso

(Bacharel em Ciência da Computação) - Universidade do Extremo Sul Catarinense, Criciúma, 2020.

OLIVEIRA, F.; LINS, L.; BARRETO, A.; ARAÚJO, J. Investigação dos efeitos do envelhecimento de software na plataforma Docker. In: Workshop em Desempenho de Sistemas Computacionais e de Comunicação, 2019. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, 2019.

REDHAT. **O que é virtualização?** 2019. Disponível em: https://www.redhat.com/pt-br/topics/virtualization/what-is-virtualization Acesso em: 16 mar. 2023.

REIS, D.C. dos S. **Execução e gestão de aplicações conteinerizadas**. 2017. 169f. Dissertação (Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos) - Faculdade de Ciências da Universidade do Porto. Porto, 2017.

ROOS, C.; SARTORI, S.; PALADINI, E.P. Uma abordagem do Lean Office para reduzir e eliminar desperdícios no fluxo de valor de informações e conhecimentos. In: Encontro Nacional de Engenharia de Produção, 31, 2011. Belo Horizonte. **Anais** [...]. Rio de Janeiro: ABEPRO, 2011.

SCHAEFER, A.; NARDI, J.C. Suporte ao desenvolvimento de software baseado em metodologia ágeis e ferramentas case em nuvens. **Revista Ifes Ciência**, [*s.l.*], v.6, n.2. p.207-227, 2020.

SILVA, F. H. R. **Avaliação de desempenho de contêineres Docker para aplicações do Supremo Tribunal Federal**. 2017. 80 f. Monografia (Licenciatura em Computação) - Universidade de Brasília — UnB, Instituto de Ciências Exatas Departamento de Ciência da Computação, Brasília, 2017**.**

TANENBAUM, A.S. **Redes de computadores.** 4. ed. Rio de Janeiro: Elsevier, 2003.

TURNBULL, J. **The Docker Book**. Creative Commons, 2014.

VITALINO, J. F. N.; CASTRO, M. A. N. Descomplicando o Docker. 2. ed. Rio de Janeiro: Brasport, 2018.

VMware. **Virtualização.** 2019. Disponível em: https://www.vmware.com/br/solutions/virtualization.html Acesso em: 16 mar. 2023.