Análise de Uso da Tecnologia de Software Docker Aplicando Conteinerização na Computação em Nuvem

Herick Maciel Maia[1], Paulo João Martins[1]

[1]Ciência da Computação – Universidade do Extremo Sul de Santa Catarina (UNESC) – Criciúma – SC – Brasil

herick9898@unesc.net, pjm@unesc.net

Abstract. *In the current computational world, it is common, for example, in a software development team, to have problems in the configuration of environments, in this way, there are tools that can facilitate the work of the developer, such as Docker. In order to apply the containerization proposed by Docker, in order to demonstrate its efficiency, a website prototype was developed in order to verify how it would be deployed with this technology, performing tests with the execution of this containerization tool. It was concluded after the production of the project, the advantages of using Docker, within the technological advances of computing.*

Resumo. *No atual mundo computacional, é comum, por exemplo, em uma equipe de desenvolvimento de software, haver problemas na configuração de ambientes, desta forma, surge ferramentas que possam facilitar o trabalho do desenvolvedor, tal como o Docker. Com o objetivo de aplicar a conteinerização proposta pelo Docker, a fim de demonstrar sua eficiência, foi elaborado um protótipo de Website em busca de verificar como seria seu deploy com esta tecnologia, efetuando testes com a execução desta ferramenta de conteinerização. Concluiu-se após a produção do projeto, as vantagens de se utilizar Docker, dentro dos avanços tecnológicos da computação.*

1. Introdução

Não há dúvidas do avanço das tecnologias por toda a sociedade, sendo responsáveis por facilitar diversas questões de nossas vidas. Modificando o cotidiano e comportamento das pessoas, caracteriza-se pela inovação de informação e da rápida e crescente comunicação [Alves 2009].

No que se refere ao desenvolvimento de software, levando em conta o avanço tecnológico, no contexto da computação em nuvem, vem surgindo ferramentas que possam facilitar o trabalho do desenvolvedor, tal como o Docker. É comum, por exemplo, haver dificuldades para configurar ambientes de desenvolvimento mesmo que sejam de configurações simples, principalmente, quando um novo membro é integrado ao time [Silva 2017].

Na tecnologia para nuvem, nota-se seus vastos benefícios, como compatibilidade, facilidade de uso, segurança e escalabilidade, assim retirando maiores complicações tecnológicas. Tendo em vista essa gama de positividades, é totalmente plausível pensar em buscar tais interesses, pois, investir em novas tecnologias, traz vantagens operacionais e melhorias reconhecidas para os negócios [Spagnuolo e Silva e Costa 2017].

Levando em consideração o potencial da computação em nuvem aliado ao desenvolvimento de software, foi escolhido utilizar a ferramenta Docker, como uma plataforma capaz de suprir as dificuldades ao se configurar um ambiente responsável por iniciar uma determinada aplicação da mesma maneira, seja em uma máquina ou em outra.

Usada não apenas por empresas que precisam crescer rapidamente seu poder computacional para atender aos requisitos dos seus negócios, mas também para diminuir os recursos, quando necessário, cabe uma orientação para identificação da real necessidade de adoção dos recursos da computação em nuvem [Arruda 2013].

Foi desenvolvido então, um protótipo de Website, com algumas tecnologias em ascensão no mercado atual de desenvolvimento, para demonstrar como seria sua inicialização com a tecnologia Docker, onde sua configuração estaria toda arquitetada de modo automatizado, retirando maiores dificuldades na entrega de um sistema.

2. Conteinerização

A computação em nuvem, pela possibilidade de ser aplicada na resolução de problemas do cotidiano, traz agilidade e produtividade no cenário moderno, devido às tecnologias constantes que surgem de forma a facilitar processos no universo profissional.

No mundo da virtualização o conceito de contêineres não é novo, mas tem adquirido maior relevância com a crescente adoção do sistema Docker [Nickoloff e Kuenzli 2019]. Para muitas empresas, esta ferramenta torna-se fundamental, com seu comportamento previsível, portabilidade e de caráter ágil.

Contêineres são rápidos, eficientes e compactos, através de suas funcionalidades e benefícios, trabalhando com múltiplas tarefas, além de que, em resposta a falhas, podem ser reiniciados rapidamente ou ser executado por longos períodos, com sua alta resiliência [Fernandez 2018].

A conteinerização, por meio da virtualização de recursos a nível de sistema operacional, possibilita a utilização de um mesmo hospedeiro físico por dois ou mais usuários ou aplicações, se comparada, por exemplo, com virtualização baseada em *hypervisor*, contêineres, destacam-se na maioria dos aspectos [Silva 2017].

Como a virtualização baseada em *hypervisor* emula dispositivos de hardware, isso acarreta um alto uso de processamento, o que torna-o uma opção muito mais pesada para um simples *deploy* de uma determinada aplicação, desta forma a conteinerização se torna mais acessível aos serviços necessários para configuração de um ambiente de desenvolvimento.

Trabalhando em cima de uma aplicação e suas dependências, o contêiner não cria uma virtualização de toda a máquina, desta forma, se obtém o benefício de rodar ambientes isolados dentro de uma única Virtual Private Server (VPS). Além, do benefício sobre o tamanho do sistema virtual, pois só integrará as configurações da sua aplicação [Dua e Raja e Kakadia 2014].

Para que os desenvolvedores possam ter uma resposta mais concreta de suas atividades, a integração do Docker com o conceito de computação em nuvem promove a expansão dos projetos, já que tal ferramenta possibilita trabalhar com diversos ambientes, constituindo-se a conteinerização um recurso simples e rápido de uso.

Os contêineres são compreendidos como uma estratégia de virtualização em nível de sistema operacional. Técnica essa baseada no isolamento de recursos específicos do kernel para que as aplicações tenham a impressão de estarem executando em um ambiente isolado [Alles 2018].

Conhecido como um método moderno de virtualização de aplicações, os contêineres, garantem um isolamento de outros processos computacionais, além de prover a portabilidade das aplicações, diferente da virtualização com *hypervisor* que existe uma camada que fornece os recursos para a virtualização de outros sistemas operacionais, na conteinerização os recursos são compartilhados com o sistema operacional principal e fornecidos para os ambientes virtualizados [Barbosa 2018]. A Figura 1 apresenta essa diferença entre as tecnologias citadas.

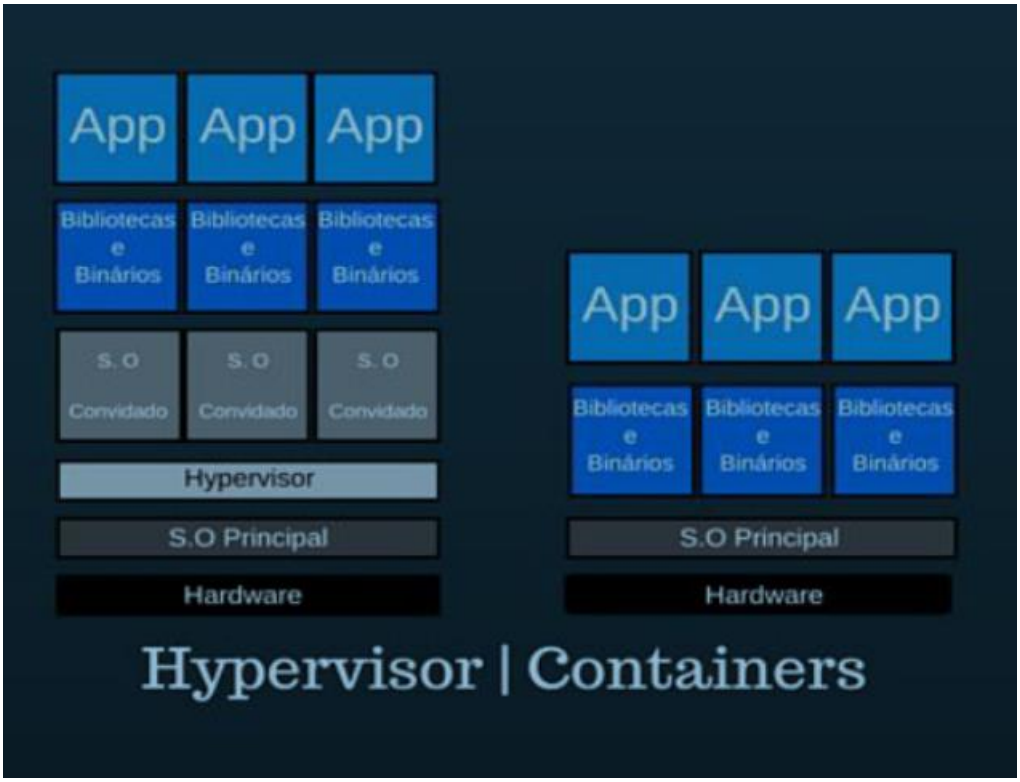

Figura 1. Comparativo entre virtualização e contêiner

Fonte: Silveira (2018)

Abstraindo para contêiner, entende-se, como uma instância de imagem em execução. Uma imagem, nesse caso, representa um ambiente virtualizado em um formato que possa ser salvo em algum meio de armazenamento, para que esse mesmo ambiente possa ser reproduzido múltiplas vezes. Proporcionando uma maneira padrão de empacotar código, configurações e dependências de uma determinada aplicação em único objeto, o contêiner faz jus a tecnologia, facilitando processos [Alles 2018].

A tecnologia propagadora de conteinerização, é o Docker, tecnologia essa, utilizada no conceito de desenvolvimento de softwares atual, aliada à cultura DevOps. Com imagens que se encontram em seu repositório de imagens oficial, o Docker Hub, surge como solução de virtualização baseada em contêiner [Dockerhub 2020].

3. Materiais e Métodos

A técnica de simulação foi aplicada no presente artigo, com o intuito de demonstrar a eficiência da tecnologia de software Docker, em relação ao método tradicional de *deploy* no contexto de desenvolvimento de software. Esta técnica foi escolhida por utilizar interações diretas com os comandos executados, de custo gratuito.

Utilizando métricas de performance, pode-se verificar confiabilidade, disponibilidade e velocidade. Para a construção deste projeto, foram definidas uma métrica para disponibilidade e velocidade na entrega de um protótipo de Website.

Para realizar os testes necessários que compõem a análise, foi utilizado a ferramenta Apache JMeter na versão 5.3. Esta ferramenta é desenvolvida na linguagem Java para avaliar o comportamento funcional de carga e medir o desempenho. Originalmente projetada para testar aplicativos Web, e expandido para outras funções de teste. Neste trabalho, foi aplicada para mensurar o tempo de execução de comandos, que realizam a entrega de um protótipo de um sistema Web.

O protótipo de Website citado, foi construído em Python na versão 3.6, Django na versão 3.0 e PostgreSQL em sua versão Alpine, sendo capaz de reproduzir quatro operações básicas, como a adição, leitura, edição e remoção de dados, então assumindo o papel do desenvolvedor que necessita testar seu software, realizando seu *deploy*, realizou-se um teste de modo automatizado com Docker e em seguida de modo manual, dentro de um servidor na nuvem da Amazon, com as seguintes configurações:

a) máquina: docker01;
b) memória: 1 GB;
c) vCPU: 2;
d) armazenamento: 20 GB;
e) SO: Linux/UNIX.

Foi mensurado o tempo de execução do *deploy* de um sistema Web, com o JMeter, por meio do comando Docker “docker-compose up --build -d”, responsável por montar a imagem contida dentro do arquivo “Dockerfile”, e então abstrair dois contêineres, um responsável pelo banco de dados, e o outro pelo sistema, fazendo com que o software seja iniciado da forma que foi estruturado para ser executado.

Realizou-se 20 testes por meio do comando citado e então tirou-se uma média do tempo resultante que o JMeter entrega. Esta mesma metodologia foi aplicada para forma de *deploy* tradicional, onde a soma de tempos dos comandos executados nos entrega um resultado.

Possibilitar que desenvolvedores trabalhem com o mesmo ambiente de produção, porém, em suas próprias máquinas, é possível, desde que o Docker esteja instalado para fazer uso de uma ou mais imagens que executam determinado sistema. A fim de demonstrar este fluxo, foi elaborada a Figura 2.

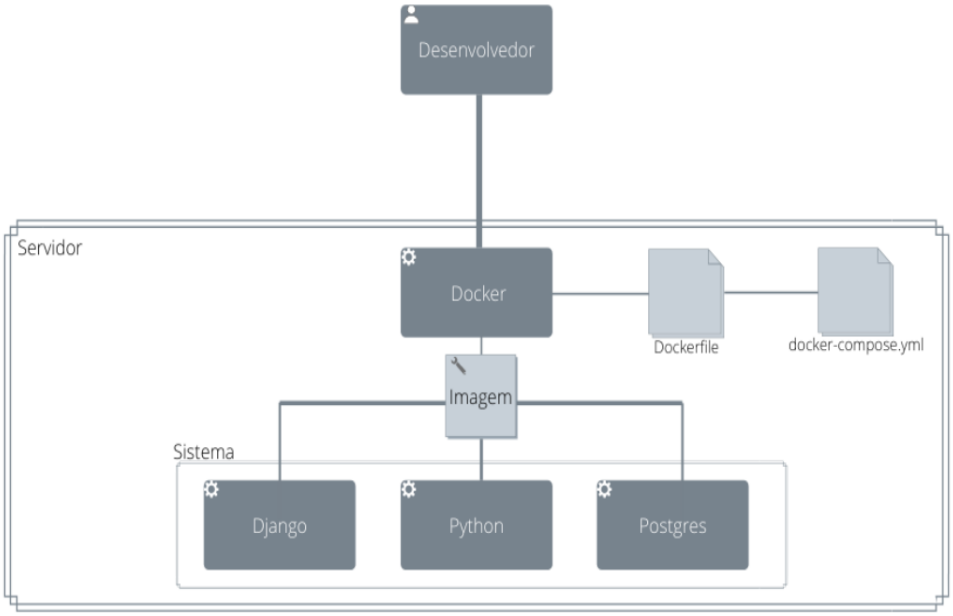

Figura 2. Fluxo de interação do desenvolvedor com o Docker

O desenvolvedor encaminha uma simples informação ao Docker, sugerindo para que seja executado o *deploy* do sistema, desta forma, esta ferramenta se responsabiliza de realizar o restante, utilizando os arquivos "Dockerfile" e o "docker-compose.yml" para construir uma imagem responsável por conter o que é necessário para a execução do sistema.

3.1. Docker

Conforme é abordado no mundo computacional, a tecnologia de software Docker, é utilizada no cenário atual de desenvolvimento de software, onde se alinha com à cultura DevOps, podendo entregar o *deploy* de determinado sistema desenvolvido.

Docker então, é apresentado como uma plataforma de código aberto usada para o desenvolvimento, distribuição e execução de aplicações, possibilitando a criação de contêineres.

3.1.1. Dockerfile

Dockerfile pode ser entendido como um arquivo de texto simples, composto por uma sequência de comandos, semelhante a uma "receita" para a criação de imagens, imagem essa que pode ser entendida como a base da criação de contêineres, composta por todo o conteúdo de um contêiner, sendo então a única fonte de informação necessária para inicializar uma aplicação [Vaz 2017].

A fim de mostrar alguns procedimentos, a Figura 3 apresenta, o conteúdo do "Dockerfile" construído para este projeto.

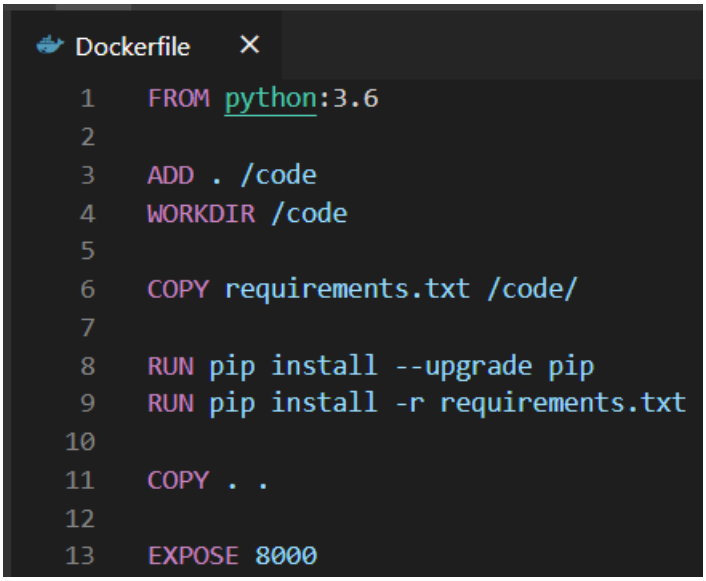

Figura 3. Estrutura do Dockerfile

É normalmente utilizado, referenciar uma imagem existente (imagem base) e modificá-la para a construção de uma nova imagem com aplicações, dependências e configurações adicionais, este processo é feito por meio do arquivo "Dockerfile". Na primeira linha mostra qual é a imagem base, a partir da qual uma nova imagem será construída. Na ilustração, a imagem base *python:3.6* define que as bibliotecas e binários disponíveis no contêiner serão as mesmas de uma linguagem Python na versão 3.6. A linha 3 informa que todo o conteúdo do diretório corrente (".") deve ser adicionado para o diretório "/code" do contêiner. A linha 4 define qual será o diretório de trabalho. Nas seguintes linhas é copiado o arquivo "requirements.txt", que contém pendências a serem instaladas, como o Django e a biblioteca do PostgreSQL, chamada Psycopg2, em sua versão 2.8, para então executar em seguida, com o comando "RUN", e então a porta 8000 é exposta, para a aplicação ser acessada.

3.1.2. Docker Compose

Docker Compose possibilita a criação conjunta de grupos de contêineres para um propósito comum a partir de um script [Vaz 2017]. Apresenta-se como uma ferramenta que define e executa múltiplos contêineres de aplicações Docker. Esta ferramenta, utiliza-se um arquivo YAML, para configurar os serviços de suas aplicações. Então, com um único comando, é criado e iniciado todos os serviços a partir de sua configuração. Docker Compose trabalha em todos ambientes, seja de produção, teste ou desenvolvimento [Docker 2020].

A Figura 4 apresenta o conteúdo do arquivo "docker-compose.yml" utilizado neste projeto, com algumas funções. Em um primeiro momento é definido a versão a ser utilizada da ferramenta, logo após vem os serviços, onde na ilustração, são apresentados dois, uma aplicação Web e um banco de dados. Para o serviço definido como "database", é nomeado um contêiner, "container_db", e então é definido uma imagem base, "postgres:alpine", seguida de suas variáveis de ambiente, e ao fim a porta que ficará exposta. No serviço chamado "web", o Docker abstrai um contêiner nomeado "container_web", e então utiliza-se uma imagem construída com "Dockerfile", no diretório corrente ("."), como por exemplo o da Figura 3, em seguida é disparado apenas

dois comandos, sendo um responsável por formular o banco de dados e o segundo para iniciar o software na porta 8000. Em “volumes”, é possível fazer uma conexão com o código fonte do protótipo, com o que fica dentro do contêiner, desta forma, toda alteração feita em uma das partes, será entendida tanto por uma quanto por outra. Ao fim é exposta a porta onde a aplicação será executada, e feito uma dependência com o serviço “database”.

docker-compose.yml

```yaml
version: '3'

services:
  database:
    container_name: container_db
    image: postgres:alpine
    environment:
      - POSTGRES_DB=postgres_db
      - POSTGRES_USER=postgres_user
      - POSTGRES_PASSWORD=postgres_pass
    ports:
      - '5432'
  web:
    container_name: container_web
    restart: always
    build: .
    command: >
            bash -c "python3 manage.py migrate
            && python3 manage.py runserver 0.0.0.0:8000"
    volumes:
      - .:/code
    ports:
      - "8000:8000"
    depends_on:
      - database
    links:
      - database:database
```

Figura 4. Estrutura do docker-compose.yml

3.1.3. Configuração do Banco de Dados PostgreSQL

Foi possível construir um banco de dados em PostgreSQL, com o auxílio do Python e Docker, apenas para persistir o protótipo durante sua execução, onde sua inicialização foi feita através dos arquivos “Dockerfile” e “docker-compose.yml”, conforme Figura 3 e Figura 4, referenciando o comando “python3 manage.py migrate” que cria um banco de

dados a partir do que foi estruturado dentro do código fonte do protótipo, sendo necessário apenas alterar o arquivo “settings.py” do projeto, para seu efetivo funcionamento.

A Figura 5 mostra as linhas, do arquivo “settings.py”, que faz referência ao contêiner executado pelo “docker-compose.yml”, com suas variáveis necessárias para execução.

Figura 5. Arquivo “settings.py” do protótipo

Docker foi escolhido então, primeiramente por atender esta necessidade de definir um padrão aos softwares que são desenvolvidos, seja em ambientes de produção ou desenvolvimento. Com sua facilidade de uso, se encaixa no contexto de desenvolvimento de software, cumprindo seu objetivo no meio computacional. Conforme foi citado na introdução, é uma plataforma capaz de suprir as dificuldades ao se configurar um ambiente responsável por iniciar uma determinada aplicação da mesma maneira, seja em uma máquina ou em outra.

3.2. Análise

Para obter os resultados dos testes citados, foram observadas duas métricas, a velocidade da criação do ambiente com o Docker, e do modo tradicional. O teste efetuado visou mensurar o tempo de *deploy* do protótipo de Website.

Para realizar este processo com o Docker, foi transferido o protótipo construído, para o servidor na nuvem citado anteriormente, e então, com o Docker instalado no servidor, foi executado o comando “docker-compose up --build -d” 20 vezes, comando este responsável por executar o arquivo “docker-compose.yml” apresentado na Figura 3, que depende do arquivo “Dockerfile” apresentado na Figura 2, para construir o ambiente necessário para executar o sistema em questão, ressaltando, que antes de cada nova execução do comando em Docker, foi realizado a exclusão de todas as imagens e contêineres que o comando “docker-compose up --build -d” cria, desta forma, o JMeter entregou o tempo de cada execução, para realizar a média da velocidade da criação do ambiente com o Docker. Este mesmo teste foi efetuado no modelo tradicional de *deploy*, instalando manualmente no servidor na nuvem os softwares Python, Django e Postgres apresentados no “docker-compose.yml” e “Dockerfile”, cada comando de instalação apresentou um tempo de execução entregue pelo JMeter, onde em sua soma representou

o tempo final para o teste, desta forma, todos esses softwares e pendências, necessários para a execução do protótipo, foram desinstalados do servidor, de forma a realizar um novo teste, realizando estes processos com o modelo tradicional também por 20 vezes.

4. Resultados e Discussões

A fim de mostrar os testes efetuados com a ferramenta JMeter, com o Docker, e de modo manual, a Tabela 1 nos mostra estes dados, onde a primeira coluna informa em um critério numérico qual teste está sendo realizado, a segunda coluna apresenta quantos milissegundos levou cada teste, e a última coluna mostra o tempo final em segundos.

Tabela 1. Testes realizados com JMeter

Testes com Docker			Testes Manuais		
Teste	Milissegundos	Tempo Final	Teste	Milissegundos	Tempo Final
1	39789	00:00:40	1	72685	00:01:13
2	39308	00:00:39	2	71850	00:01:12
3	39456	00:00:40	3	71073	00:01:11
4	40389	00:00:40	4	71130	00:01:11
5	38873	00:00:39	5	70847	00:01:11
6	39401	00:00:39	6	71141	00:01:11
7	39588	00:00:40	7	71274	00:01:11
8	39720	00:00:40	8	71476	00:01:12
9	39363	00:00:39	9	70994	00:01:11
10	39844	00:00:40	10	70890	00:01:11
11	39135	00:00:39	11	71754	00:01:12
12	38607	00:00:39	12	71964	00:01:12
13	39303	00:00:39	13	71541	00:01:12
14	39027	00:00:39	14	74089	00:01:14
15	39585	00:00:40	15	71806	00:01:12
16	39355	00:00:39	16	71682	00:01:12
17	40183	00:00:40	17	72517	00:01:13
18	39083	00:00:39	18	74114	00:01:14
19	39224	00:00:39	19	72091	00:01:12
20	39174	00:00:39	20	71974	00:01:12

Foi observado que o protótipo de Website, construído em Python, Django e Postgres teve um *deploy* mais vantajoso com a ferramenta de software Docker, apresentando um tempo de execução do sistema, mais eficiente do que comparado ao modo manual.

A Tabela 2 mostra a média dos tempos obtidos de cada modelo de execução do protótipo a partir da Tabela 1.

Tabela 2. Resultados dos testes

Teste	Milissegundos	Tempo Final
Docker	39420,35	00:00:39
Manual	71844,60	00:01:12

Apesar da média do tempo de execução em ambos modelos, terem sido relativamente próximos, é plausível pensar que em um sistema mais complexo, com diversas pendências e softwares necessários para sua execução, o Docker se apresente mais eficiente, de modo que a instalação de forma manual dessas pendências seria mais trabalhosa.

Vale ressaltar também, a automação que o Docker entrega, em comparação com o método de *deploy* tradicional, tendo em vista que com poucas ações todo um ambiente é iniciado para a execução de um software, se aliando a flexibilidade, com sua capacidade de realizar alterações de forma mais ágil na padronização dos projetos.

Tendo em vista esta agilidade entregue pelo Docker, é perceptível que esta ferramenta supre as dificuldades ao se configurar um ambiente de uma determinada aplicação, desta forma, um sistema pode ser executado da mesma maneira, seja em uma máquina ou em outra.

O trabalho realizado por Vaz (2017), que apresentou uma comparação entre métricas de um modelo manual de *deploy* e uma solução de modelo baseado em uma técnica de virtualização, apontou resultados bastante significativos, levando em consideração que o número de ferramentas e pendências a serem configurados pelos testes em Docker e de modo manual, foi maior que neste presente artigo, onde à velocidade da criação do ambiente virtual se mostrou mais eficiente com o Docker, precisando de menos do que cinquenta minutos, já no teste feito manualmente, todo o processo levou cerca de uma hora e trinta minutos. Mostrando a notável diferença de performance entre o modelo tradicional de *deploy* e o modelo com o Docker, sendo este último uma solução muito mais ágil e eficiente.

A pesquisa apresentada por Silva (2017), tendo o objetivo de realizar uma avaliação de desempenho do uso de contêineres Docker, efetuando testes em uma aplicação da Secretaria de Tecnologia da Informação do Supremo Tribunal Federal, entre ambientes de conteinerização, apontou resultados demonstrando que o desempenho do ambiente com utilização de contêineres foi superior ao nativo, também comprovou que, o tempo de resposta da aplicação foi similar, tanto com o uso do Docker como no modelo tradicional, afirmando não haver deterioração de performance da aplicação com o uso da plataforma de contêineres.

No artigo elaborado por Silveira (2018), o autor demonstra superioridade do Docker, em comparação a uma outra ferramenta que cria ambientes virtuais de testes e desenvolvimento, o Vagrant. Apresentando o Docker, como uma ferramenta completa de conteinerização, possuindo diversas funcionalidades nativas e um desempenho melhor na questão de *deploy*, versionamento e gerenciamento de contêineres, com muita mais documentação disponível, porém, também apontou um aspecto positivo do Vagrant, onde esta ferramenta permite a utilização de contêineres com sistemas operacionais diferentes.

Assim como apresentado nos trabalhos correlatos, neste presente artigo, tendo em vista as análises e resultados descritos, o Docker também se mostra eficiente, em relação ao modelo tradicional de *deploy*, com suas diversas características apontadas, tem crescido constantemente no mundo computacional [Ufsm 2017].

5. Conclusão

Com o avanço da tecnologia no mundo profissional, a adoção de metodologias modernas, a fim de aumentar a eficiência da produção de software e beneficiar a relação entre as diferentes áreas da computação, é totalmente plausível. A conteinerização em ambientes na nuvem, contribui na solução de diversos problemas de desenvolvimento de software.

Este artigo visou demonstrar a utilização da tecnologia de software Docker, comparando com o modelo tradicional de *deploy*, apontando resultados significativos dentre a análise apresentada, mostrando sua eficiência de automação e flexibilidade. De fato, esta tecnologia tende a aparecer com maior frequência conforme vai mostrando sua capacidade no mundo computacional, levando em consideração que é necessária uma identificação da real adoção dos recursos da computação em nuvem.

As dificuldades encontradas foram durante o processo de estudo da ferramenta escolhida, tendo em vista que apesar de sua ascensão, é uma tecnologia nova que requer aprendizados para sua devida utilização, porém, o objetivo principal, aplicar a conteinerização no contexto de desenvolvimento de software, por meio da tecnologia Docker, foi alcançado após a realização dos testes e a obtenção dos resultados.

Para dar continuidade aos estudos, sugere-se como realização de trabalhos futuros, testes com o Docker em um protótipo de software com maior diversidade de ferramentas e pendências, visando comprovar sua efetividade em um outro cenário. Outro critério a ser levantado, seria a aplicação da conteinerização em um ambiente empresarial, mostrando sua utilização.

Referências

ALLES, Guilherme Rezende. Análise da utilização de tecnologias de contêineres para aplicações de alto desempenho. 2018. 59 f. Monografia (Especialização) - Curso de Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2018.

ALVES, Taíses Araújo da Silva. Tecnologias de informação e comunicação (tic) nas escolas: da idealização à realidade: estudos de casos múltiplos avaliativos realizado em escolas públicas do ensino médio do interior paraibano brasileiro. 2009. 134 f. Dissertação (Mestrado) - Curso de Ciências da Educação, Universidade Lusófona de Humanidades e Tecnologias, Lisboa, 2009.

ARRUDA, Eduardo Nascimento de. Identificando o grau de dependência da adesão à computação em nuvem. 2013. 208 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Universidade Federal de Pernambuco, Recife, 2013.

BARBOSA, Hefrayn Antero Barros. Análise das vantagens do uso de CI/CD no desenvolvimento de um projeto java web. 2018. 64 f. TCC (Graduação) - Curso de Engenharia da Computação, Centro Universitário de Anápolis, Anápolis, 2018.

DOCKER, 2020. Disponível em: <https://docs.docker.com/compose/>. Acesso em: set. 2020.

DOCKERHUB, 2020. Disponível em: <https://hub.docker.com/>. Acesso em: set. 2020.

DUA, R; RAJA, A R; KAKADIA, D. Virtualization vs Containerization to Support PaaS. 2014, [S.l: s.n.], 2014. p. 610–614.

FERNANDEZ, Gabriel Pereira. Orquestração de Contêineres na Nuvem: um modelo de segurança. 2018. 105 f. Dissertação (Doutorado) - Curso de Ciência da Computação, Universidade Federal de Campina Grande, Paraíba, 2018.

NICKOLOFF, J; KUENZLI, S. Docker in Action. [S.l.]: Manning Publications, 2019. Disponível em: <https://books.google.com.br/books?id=bmVwwQEACAAJ>.

SILVA, Flávio Henrique Rocha e. Avaliação de desempenho de contêineres Docker para aplicações do Supremo Tribunal Federal. 2017. 80 f. Monografia (Especialização) - Curso de Ciência da Computação, Universidade de Brasília, Brasília, 2017.

SILVEIRA, Gustavo Ferraz. Comparativo de desempenho e funcionalidades entre Docker e Vagrant. Pelotas: Senac, 2018.

SPAGNUOLO, Fernando de Oliveira; SILVA, Murilo Helderson Martins; COSTA, Willian Maciel. A importância da tecnologia da informação no suporte à tomada de decisões. 2017. 57 f. TCC (Graduação) - Curso de Administração, Centro Universitário Católico Salesiano, Lins, 2017.

UFSM, 2017. Disponível em: <https://www.ufsm.br/pet/sistemas-de-informacao/2017/03/30/o-docker-e-seus-containers-a-nova-era-da-virtualizacao/>. Acesso em: nov. 2020

VAZ, João Victor Uchôa. GenCloud - Uma abordagem para disponibilização de sistemas web baseado em contêineres em um ambiente de nuvem híbrida. 2017. 66 f. TCC (Graduação) - Curso de Sistemas de Informação, Universidade Federal de Pernambuco, Recife, 2017.