INFRAESTRUTURA PAAS PARA UM AMBIENTE SEGURO E ESCALÁVEL DA PLATAFORMA RETRATOS DA ATENÇÃO PRIMÁRIA À SAÚDE

Eduardo Jorge Valadares Oliveira
Doutor e Mestre em Engenharia Biomédica pela Universidade Estadual de Campinas. Graduação em Engenharia Elétrica pela Universidade de Alfenas.. Professor do Núcleo de Tecnologias Estratégicas em Saúde (NUTES/UEPB) da Universidade Estadual da Paraíba e Professor Visitante da Universidade da Califórnia em San Diego (UCSD).
E-mail: edujvo@servidor.uepb.edu.br

Nadja Naira Valente Mayrink
Doutoranda em Estudos Contemporâneos no Centro de Investigação Interdisciplinar (CEIS20) da Universidade de Coimbra - Portugal. Especialização em Economia da Saúde pela Universidade de São Paulo - USP. Graduação em Pedagogia pela Universidade de Brasília - UNB. Pesquisadora do Núcleo Avançado de Inovação Tecnológica - NAVI - IFRN.
E-mail: nadja.mayrink@lais.huol.ufrn.br

Alex Fabiano de Araújo Furtunato
Mestrado em Engenharia Elétrica pela Universidade Federal do Rio Grande do Norte. Graduação em Engenharia Elétrica pela Universidade Federal do Rio Grande do Norte Pesquisador do Núcleo Avançado de Inovação Tecnológica (NAVI/IFRN).
E-mail: alex.furtunato@ifrn.edu.br

Welkson Renny de Medeiros
Mestrado Profissional em Engenharia de Software pela Universidade Federal do Rio Grande do Norte. Especialização em Sistemas Corporativos pelo Centro Universitário do Rio Grande do Norte, UNIRN. Graduação em Sistemas de Informação pela Universidade Potiguar. Pesquisador do Núcleo Avançado de Inovação Tecnológica (NAVI/IFRN). E-mail: welkson.medeiros@ifrn.edu.br

RESUMO

A computação em nuvem surgiu como uma solução para diversos problemas de disponibilidade e resiliência de aplicações na internet. Novos conceitos surgiram com esse novo paradigma como, por exemplo, "conteinerização" de aplicações, infraestrutura como código, microsserviços, entre outros. Este trabalho propõe um novo ambiente de infraestrutura para os sistemas do ecossistema de projetos do Núcleo Avançado de Inovação Tecnológica – NAVI e tomando como estudo de caso o sistema Retratos da Atenção Primária à Saúde. O novo ambiente é baseado em uma solução de nuvem computacional conhecida como PaaS. É proposta uma nova estratégia de implantação de aplicações utilizando o OpenShift com scripts de CI/CD, permitindo aos desenvolvedores um maior controle no processo de integração contínua e de implantação dos seus projetos. Além das vantagens de gerenciamento de implantações, os resultados obtidos mostram que o novo ambiente proposto também traz ganhos consideráveis de performance das aplicações e nova possibilidades de escalonamento horizontal das aplicações. Nos testes realizados, foi obtido um ganho mínimo de 13% de performance.

Palavras-chave: Infraestrutura. Devops. Docker. Orquestração de containers. OpenShift.

ABSTRACT

Cloud computing has emerged as a solution to several problems of availability and resilience of applications on the internet. New concepts have also emerged with this new paradigm: applications containerization, infrastructure as code, and microservices. This work proposes a new infrastructure environment for the ecosystem systems of the "Núcleo Avançado de Inovação Tecnológica – NAVI" and take was a case study the system Retratos da Atenção Primária à Saúde. The computing cloud solution known as PaaS is the base for the new environment. We proposed a new application deployment strategy using OpenShift with CI / CD scripts, allowing developers greater control over the process of continuous integration and deployment of their projects. In addition to the benefits of managing deployments, the results obtained show that the proposed environment also brings considerable gains in application performance and new possibilities for horizontal scaling of applications. In the tests performed, we got a minimum gain of 13% in the system performance..

KEYWORDS: Infrastructure. Devops. Docker. containers orchestration. OpenShift.

INTRODUÇÃO

Em uma sociedade cada vez mais conectada e centrada em dados, os sistemas de informação tornam-se essenciais para o bom funcionamento dos mais variados serviços fornecidos por órgãos públicos, corporações privadas, entre outros. A informatização dos serviços proporciona, por exemplo, que os cidadãos possam ter acesso a serviços públicos de maneira mais eficiente e informações que antes eram praticamente inacessíveis. Além disso, a facilidade de acesso através de computadores e, principalmente, de dispositivos móveis, traz uma maior comodidade para os usuários. Portanto, a indisponibilidade desses serviços pode trazer grandes transtornos para os usuários e, consequentemente, um desgaste para a imagem da instituição fornecedora.

A adoção da computação em nuvem (C*loud Computing*) tem sido uma das estratégias adotadas para lidar com os desafios que envolvem a manutenção da disponibilidade de sistemas informatizados com larga escala de uso. Dentre as diversas definições

para o termo *Cloud Computing*, uma forma concisa de definir seria "Computação em Nuvem é uma forma especializada de computação distribuída que introduz modelos de utilização de provisionamento remoto escalável e recursos medidos" [1] (ERL; PUTTINI; MAHMOOD, 2013, p. 28).

Para alcançar o objetivo da alta disponibilidade de sistemas, os grandes provedores de computação em nuvem usam diversas estratégias, tais como, localização de datacenters em diferentes regiões geográficas do globo, mecanismos para garantir a redundância do fornecimento de energia, construção de usinas geradoras de energia próprias, redundância de links de internet, entre outros.

Ao adotar uma *cloud,* o cliente deve escolher umas das modalidades de fornecimento do serviço, tais como a IaaS *(Infrastructure as a Service)*, PaaS *(Plataform as a Service)*, SaaS *(Software as a Service)*, entre outras (MELL; GRANCE, 2011). No modelo IaaS, o cliente tem a possibilidade de instalar seus aplicativos e fazer ajustes na máquina virtual, no entanto, o cliente também deve se preocupar em como configurar os serviços para que a plataforma seja elástica (se ajuste automaticamente para suportar uma maior demanda de usuários). No modelo PaaS, a plataforma de nuvem é responsável por escalar o sistema de forma automática para suportar uma maior demanda de usuários, entretanto, o desenvolvedor tem que adaptar seus sistemas ao modelo previsto pela fornecedora de nuvem (em alguns casos, requer uso de bibliotecas específicas, por exemplo). Por fim, temos o SaaS, no qual o fornecedor de nuvem é responsável pelo desenvolvimento e manutenção do software, como também da infraestrutura.

Neste trabalho, é apresentado o relato da adoção de uma PaaS no contexto de um sistema de saúde, o Retratos da Atenção Primária à Saúde, que reúne informações de mais de 42 mil equipes de saúde presentes em 5.324 municípios participantes do 3º Ciclo do Programa Nacional de Melhoria do Acesso e da Qualidade da Atenção Básica (PMAQ-AB). Disponibilizando, então, dados de 13.775 equipes de Saúde da Família, 25.090 equipes de Saúde da Família com Saúde Bucal, 4.110 Núcleos Ampliados de Saúde da Família e Atenção Básica e 30.346 Unidades de Saúde da Família. O PMAQ-AB foi instituído na Portaria GM/MS nº 1.645 (MINISTÉRIO DA SAÚDE, 2015), e tem por objetivo induzir a melhoria do acesso e da qualidade da atenção básica. Além disso, o PMAQ estabelece um padrão de qualidade que possibilita uma comparação das ações governamentais direcionadas à Atenção Básica em Saúde em nível nacional, regional e local, e consequentemente, amplia a efetividade e transparência dessas ações.

O relato descreve o ambiente atual do sistema Retratos da Atenção Primária à Saúde usando containers no modo *standalone* com *Docker,* e suas limitações. Ademais, apresenta a proposta de uma nova arquitetura utilizando a plataforma de orquestração de containers OKD (*OpenShift Origin*) e um conjunto de scripts para execução do sistema em um modelo PaaS escalável.

O artigo é organizado da seguinte forma: a seção Referencial Teórico apresenta alguns conceitos importantes para a compreensão do trabalho; em seguida, a seção Ambiente e Configuração apresenta a proposta do ambiente de implantação dos sistemas, a sua implementação e configuração. Posteriormente, na seção Metodologia, é apresentado o plano de testes para medir o teste de carga no ambiente proposto e no ambiente legado. Após isso, na seção Resultados, são apresentados os resultados da medição descrita na seção anterior. Por fim, na seção Conclusão, é apresentado um resumo da proposta e sugestões de trabalhos futuros.

[1] Cloud computing is a specialized form of distributed computing that introduces utilization models for remotely provisioning scalable and measured resources

REFERENCIAL TEÓRICO

Nesta seção, são apresentados alguns conceitos importantes para uma melhor compreensão da proposta deste trabalho.

DevOps

DevOps integra os dois mundos, desenvolvimento e operação, automatizando o desenvolvimento, implantação e monitoramento (EBERT *et al.*, 2016). A estratégia DevOps permite a equipe de desenvolvimento e de infraestrutura trabalharem juntas para permitir que as aplicações sejam implantadas de maneira eficiente e automatizadas com a menor interferência manual. Para tal, faz-se uso de ambiente baseado em virtualização, preferencialmente containers, sistemas de gerenciamento de código fonte, assim como a mudança do paradigma de infraestrutura com a adoção do conceito de Infraestrutura como código (IaC), que é a prática de especificar e automatizar o ambiente em que um software será testado ou implantado (JIANG; ADAMS, 2015).

Docker

Container é uma unidade padrão de software que empacota o código e todas as dependências necessárias para executá-lo rapidamente e sem falhas em qualquer ambiente computacional (DOCKER INC, 2020). O Docker é um *engine* que pode ser utilizado para executar containers baseado em imagens de containers que utilizem o padrão "Docker". Ele pode ser executado em ambientes Linux e Windows permitindo, portanto, que containers possam ser executados em qualquer um desses ambientes. Containers e Máquinas virtuais possuem recursos de isolação semelhantes, mas funcionam de maneira diferente, pois, enquanto máquinas virtuais virtualizam o hardware, os containers virtualizam o sistema operacional. Portanto, containers são mais eficientes e portáveis.

Kubernetes

Kubernetes é uma plataforma de orquestração de containers (SAYFAN, 2018) *open source*, criada pelo Google, com diversas funcionalidades, tais como: montagem de sistema de armazenamento, distribuição de "secrets", crescimento escalável horizontal, balanceamento de carga, recursos de monitoramento, sistema de autenticação e autorização, entre outras. Ou seja, Kubernetes é uma plataforma para implantação de aplicações, escalável com gerenciamento baseada em containers. Enquanto no Docker, a unidade individual de execução de aplicação é denominada container, na Kubernetes, a unidade mínima de execução gerenciada é denominada POD, que pode conter um ou mais containers.

OpenShift

O OpenShift (REDHAT,2020) (DUMPLETON, 2018) é uma plataforma como Serviço (PAHL, 2015) (LOSSENT; RODRIGUEZ PEON; WAGNER, 2017) desenvolvido pela Red Hat e construída em cima do Kubernetes. Em sua versão *open source,* ele é denominado "Openshift Origin" ou OKD. Além das funcionalidades do Docker e do Kubernetes, o OpenShift adiciona várias outras funcionalidades, tais como *web console*, portal de catálogo de *templates*, automação do processo de *build* e *deployment* de aplicações, um serviço de *registry* interno e roteamento de aplicações, entre outras.

AMBIENTE E CONFIGURAÇÃO

Atualmente, os sistemas do NAVI são executados em um ambiente baseado em Docker. O Host é uma máquina virtual,

rodando no VMware vSphere ESX 5.5, cujas características técnicas estão descritas no Quadro 1. O Sistema Operacional utilizado é o CentOS 7, com Docker versão 18.09.3.

A aplicação utilizada como estudo de caso, denominada Retratos da Atenção Primária à Saúde, uma aplicação baseada no Framework Laravel e que utiliza como banco de dados o PostgreSQL versão 9.5, e é executada como container utilizando uma imagem Docker baseada no Phusion/Baseimage versão 0.11. Para o versionamento do código e dos scripts que compõem a imagem Docker, é utilizado o GitLab, instalação "On Premise". Os scripts incluídos no container, quando em execução, permitem que o sistema possa se autoatualizar a cada novo "commit" de código no repositório, mantendo assim o sistema atualizado e em sincronia com a versão do repositório, na *branch master*.

O novo ambiente proposto é composto de uma plataforma como serviço (PaaS), onde serão implantados os serviços relativos aos sistemas que compõem o arcabouço tecnológico do PMAQ. Tais serviços deverão ser implantados a partir de *templates* armazenados juntamente com os respectivos códigos fontes de cada sistema em um sistema de controle de versão (*Source Code Management* - SCM). Além disso, a proposta deverá possibilitar aos desenvolvedores das soluções o uso de práticas DevOps, através de *Continuous Integration* (CI) e *Continuous Deployment* (CD).

As ferramentas utilizadas no ambiente proposto são: para o armazenamento dos códigos fontes com versionamento dos sistemas, assim como para a execução dos *pipelines* de integração e de implantação, será mantido o *GitLab* versão "On Premise". Para a disponibilização dos sistemas, de forma "conteinerizada", e atendendo aos requisitos de implantação automatizada, será utilizado o OpenShift Community Edition, também conhecido como Origin ou OKD, versão de código aberto baseada na ferramenta RedHat OpenShift. O OKD 3.11 é a última versão estável disponível até o momento e, portanto, a que será utilizada neste trabalho.

A instalação do OKD "On premise" foi executada através do script (OKD COMMUNITY, 2019) que instala toda a infraestrutura para a ferramenta em um único Host conforme descrito no quadro a seguir o que torna essa instalação bem mais simplificada e que, nesse momento, atende bem para um ambiente de teste. Contudo, para uso em produção, a instalação do OKD deve ser feita utilizando-se uma estratégia mais robusta, composta de um número bem maior de hosts, o que possibilita um ambiente de maior disponibilidade.

Quadro 1 - Configurações dos Servidores utilizados nas medições.

Servidor Docker	Servidor OpenShift
08 CPUS Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz 32Gb Mem 100GB de disco S.O. CentOS 7 Docker, versão 18.09.3	08 CPUS Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz 32Gb Mem 100GB de disco S.O. CentOS 7 OKD, versão 3.11

Fonte: Autoria própria (2020).

A operação no ambiente baseado em Docker possui uma restrição de segurança que impede aos desenvolvedores terem acesso ao host e acompanhar o *deploy* para fazer depuração da implantação ou execução da aplicação. Essa restrição se deve ao fato de que, em um mesmo host são executados diversos containers de diferentes projetos com equipes de desenvolvimento diferentes. Portanto, o *deploy* da aplicação no host é feito pela equipe de infraestrutura, utilizando scripts próprios, baseados em imagens de containers construídas e sem a participação dos desenvolvedores do projeto nessa etapa.

Na nova proposta, os desenvolvedores terão acesso à interface web de gerenciamento de projetos do OKD, com acesso controlado por um conjunto de regras de permissão e, portanto, limitando o acesso

aos containers em execução somente aos envolvidos no projeto específico. Pela interface de gerenciamento é possível acompanhar e gerenciar todas as etapas de implantação da aplicação, com acesso aos *logs* de execução, gerenciamento de orquestração e, inclusive, com acesso de terminal aos containers em execução na aplicação.

Para um melhor gerenciamento das etapas de desenvolvimento, é proposto que para cada aplicação sejam criados, no mínimo, dois projetos no OKD: um primeiro projeto para executar a versão de desenvolvimento da aplicação (denominada aqui de "retratos-dev"), e um segundo projeto para executar a versão de produção da aplicação (denominada aqui de "retratos-prod"). A diferença entre os projetos "retratos-dev" e "retratos-prod" é a forma de implantação e de aplicação de atualizações. O diagrama de fluxo de uso dos dois projetos pode ser visto na Figura 1 a seguir.

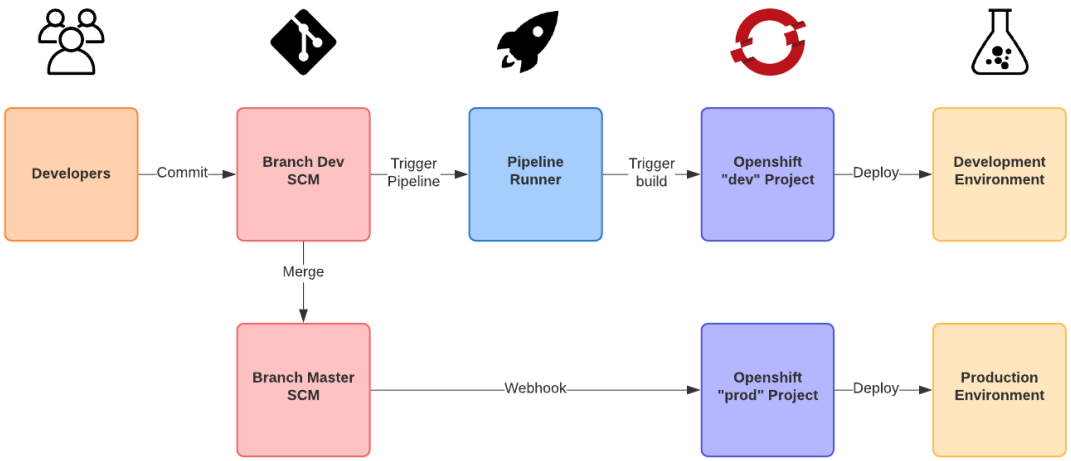

Figura 1 – Fluxo de desenvolvimento e implantação.
Fonte: Autoria própria (2020).

No projeto da versão "retratos-dev", a implantação será feita através de um *pipeline* no SCM, no qual os desenvolvedores podem escrever seus scripts de testes de segurança e testes unitários. Nessa versão, a aplicação será implantada através do código fonte armazenado em uma *branch* de desenvolvimento que, além do código fonte da aplicação também armazenará os arquivos de *templates* do OKD que serão utilizados durante a fase de *deploy* do pipeline e que definem quais objetos do Kubernetes serão criados no PaaS. Durante a implantação, os desenvolvedores poderão, portanto, realizar todos os testes e acompanhar os logs de implantação no projeto "retratos-dev" do OKD.

Para o ambiente de produção, o projeto "retratos-prod" utilizaria o código da aplicação da *branch master*. A atualização da versão de produção no projeto "retratos--prod" do OKD não será feita através de pipeline do SCM. Nesse caso, como os testes foram programados para serem executados no pipeline da versão de desenvolvimento, a versão "retratos-prod" utilizará o recurso de *webhook* do SCM, que permite ao OKD ser notificado a cada novo evento (*commit*, *merge request*, etc.) do repositório de código para que possa refazer o *deploy* da aplicação. Ou seja, após a aprovação dos testes e liberação da versão em desenvolvimento para o ambiente de produção, será solicitado um *merge request* para a *branch master* que, após ser aprovado, disparará

uma notificação via *webhook* para o OKD, na versão "retratos-prod", o que causará a execução de um novo *deploy* da aplicação, assim como reexecutar as operações necessárias, tais como *migrations* de banco de dados.

A Figura 2 mostra o projeto "retratos-prod" implantado no OKD. Pode-se observar que o desenvolvedor poderá monitorar todos os objetos que compõem o *build* e o *deploy* da aplicação. O sistema também conta com um sistema de métricas que monitoram o uso de recursos da aplicação no cluster Kubernetes gerenciados pelo OKD. É possível também aumentar o número de containers que atendem a aplicação de maneira manual, aumentando o número de "Pods", ou criar uma regra para que o sistema realize um autoescalonamento para atender aos requisitos de performance desejados.

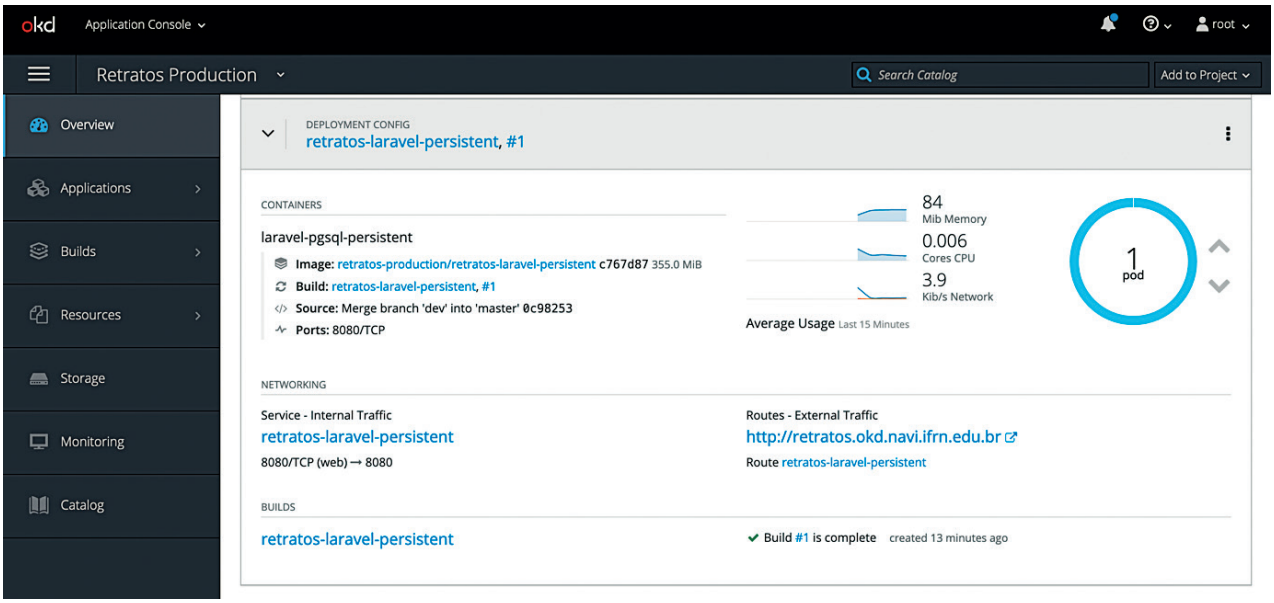

Figura 2 – OKD, Projeto "prod" implantado com a aplicação Retratos.
Fonte: Autoria própria (2020).

No Quadro a seguir é mostrado um exemplo de um script de pipeline para ambiente "retratos-dev". Nele, pode-se ter, por exemplo, um estágio "test". Além de um estágio de implantação existe também um estágio manual para realizar o "cleanup" do projeto no OKD, que permite destruir todos os objetos criados numa implantação de maneira manual, através de um botão na tela de gerenciamento de pipelines do SCM. Esse recurso manual poderia ser utilizado pelos desenvolvedores para realização de novos testes e/ou implantação.

```
# .gitlab-ci.yml
stages:
  - test
  - review
  - cleanup

variables:
  OPENSHIFT_SERVER: https://console.okd.navi.ifrn.edu.br:8443
  OPENSHIFT_DOMAIN: okd.navi.ifrn.edu.br
  OPENSHIFT_TEMPLATE_NAME: laravel-retratos

test1:
  stage: test
  before_script: []
  script:
    - echo run tests
```

```
.deploy: &deploy
  before_script:
    - oc login "$OPENSHIFT_SERVER" --token="$OPENSHIFT_TOKEN" --insecure-skip-tls-verify
  script:
    - "oc create -f ./openshift/templates/laravel-retratos.json 2> /dev/null || echo templateAlrea-
dyExists"
    - "oc get services $APP 2> /dev/null || oc new-app $OPENSHIFT_TEMPLATE_NAME --param-file=./
openshift/templates/template.params"

review:
  <<: *deploy
  stage: review
  variables:
    APP: dev-$CI_COMMIT_REF_NAME
    APP_HOST: $CI_PROJECT_NAME-$CI_ENVIRONMENT_SLUG.$OPENSHIFT_DOMAIN
  environment:
    name: review/$CI_COMMIT_REF_NAME
    url: http://$CI_PROJECT_NAME-$CI_ENVIRONMENT_SLUG.$OPENSHIFT_DOMAIN
    on_stop: stop-review
  only:
    - branches
  except:
    - master

stop-review:
  <<: *deploy
  stage: cleanup
  script:
    - oc delete all -l "app=$OPENSHIFT_TEMPLATE_NAME"
    - oc delete pvc -l "app=$OPENSHIFT_TEMPLATE_NAME"
    - oc delete template $OPENSHIFT_TEMPLATE_NAME
    - oc delete secret okd-retratos-atencao
  when: manual
  variables:
    APP: review-$CI_COMMIT_REF_NAME
    GIT_STRATEGY: none
  environment:
    name: review/$CI_COMMIT_REF_NAME
    action: stop
  only:
    - branches
  except:
    - máster
```

Fonte: Autoria própria (2020).

METODOLOGIA

Para demonstrar o cenário de uma infraestrutura do Retratos da Atenção Primária à Saúde em uma instância *Docker standalone* e no OKD com diferentes números de PODs, foram realizados testes de carga simulando requisições ao sistema-alvo, nas duas propostas de infraestruturas, com a aplicação implantada na versão "retratos-prod".

A Figura 3 apresenta uma visão do ambiente preparado para execução dos experimentos para avaliação da proposta de arquitetura resiliente do sistema PMAQ-Retratos. A VM 1 executa uma versão do "retratos" no formato legado, ou seja, uma máquina virtual *CentOS 7*, usando *Docker standalone* (sem cluster), e expondo os serviços através de containers do *NGnix*, fazendo proxy para o container da aplicação. A VM 2 executa uma instância da plataforma de orquestração de containers *OKD*. O *OKD* internamente utiliza Kubernetes, que é responsável por gerenciar o ciclo de vida dos containers (manter o número de instâncias definidas na configuração em execução, entre outros). Por fim, a VM 3 é utilizada para efetuar testes de cargas com a ferramenta *JMeter*.

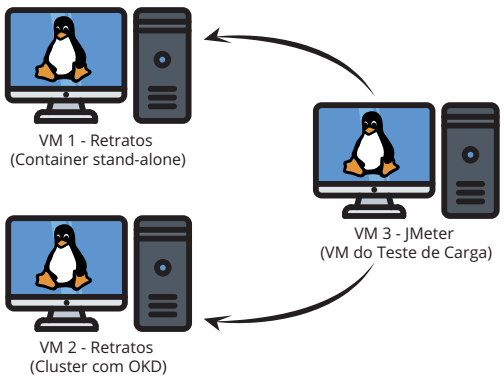

Figura 3 – Infraestrutura do Testes.
Fonte: Autoria própria (2020).

As especificações das VMs que executam o Retratos da Atenção Primária à Saúde no formato *Docker Standalone* e OKD já foram apresentadas na seção anterior. A VM do *JMeter* (VM 3) também executa um sistema operacional *CentOS 7 x64*, e tem os seguintes recursos: 4 cores, 8GB de memória RAM e 100GB de disco. A versão do *JMeter* instalado na VM3 é o 4.0 (build r1823414). As VMs estão na mesma rede local, e o tráfego gerado não passa por firewalls ou limitadores de banda para evitar que latências causadas por links ou processamento de pacotes possam comprometer os resultados dos experimentos.

RESULTADOS

Para obter os resultados do experimento, foi traçado um plano de testes que seria repetido nos dois ambientes: *Standalone* e OKD. O plano de testes foi realizado da seguinte forma: A VM 3 será utilizada para simular requisições de acesso às instâncias do Retratos da Atenção Primária à Saúde no formato legado que se encontra na VM 1, e em seguida, irá executar os mesmos testes no sistema que se encontra na VM 2. Os parâmetros utilizados no *JMeter* para execução das requisições são apresentados na tabela a seguir.

Tabela 1 - Parâmetros do JMeter.

Parâmetro	Valor
Threads	50
Ramp-Up	60
Schedule	90

Fonte: Autoria própria (2020).

O parâmetro *Threads* simula a quantidade de usuários simultâneos que estão executando requisições ao sistema. O parâmetro *Ramp-Up* é utilizado para informar o tempo total para que o *JMeter* dispare todas as threads simulando usuários simultâneos. Ou seja, no início do teste haverá apenas 1 thread em execução, e no decorrer do tempo, o *JMeter* irá disparar novas threads até chegar em 50 threads em 60 segundos, com um crescimento gradativo do número de "usuários". Após disparar todas as 50 threads, o *JMeter* ainda executará por mais 30 segundos as requisições, pois, o tempo total do teste foi definido para 90 segundos (parâmetro *Schedule*).

O objetivo do disparo gradativo de threads é avaliar como o sistema se comporta em um cenário com diferentes números de usuários simultâneos. É esperado que ocorra um maior tempo de resposta das requisições ao aumentar o número de usuários. Entretanto, em um cenário com uma infraestrutura resiliente, esses parâmetros de tempo de resposta podem ser utilizados para que o ambiente disponibilize mais recursos de forma automática, também denominada de elástica, para suportar a demanda e não causar indisponibilidade no acesso aos sistemas. Para demonstrar a escalabilidade da infraestrutura proposta e possibilitar uma melhoria no tempo de resposta, mesmo com um crescimento gradativo de usuários, o OKD também será avaliado com 03 PODs. No OKD a aplicação pode ser escalada para um número maior de PODs de maneira manual ou usando o recurso denominado *autoscaler*, que aumentará o número de PODs dependendo

de critérios de performance configurado no projeto.

Foram executados 10 (dez) testes de carga, tanto na instância Docker *standalone* quanto no OKD, na configuração com 1 (um) POD e com 3 (três) PODs, totalizando 30 execuções. A Figura 4 mostra o resultado dos testes do *JMeter* para cada instância.

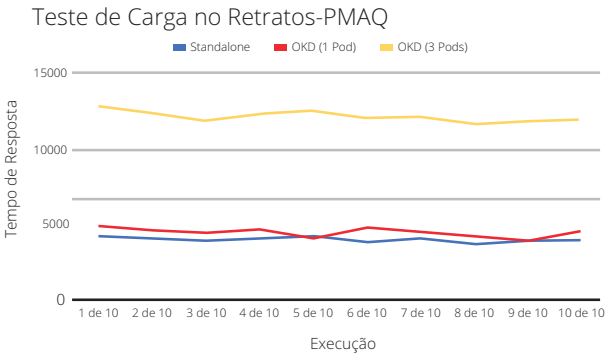

Figura 4 – Requisições atendidas em cada experimento do *JMeter.*

Fonte: Autoria própria (2020).

Para o conjunto das 10 execuções de cada instância, foi calculada a mediana da quantidade de requisições atendidas. O objetivo é diminuir o efeito de picos de processamento aleatórios, interferindo no resultado dos testes. A tabela a seguir mostra os resultados da mediana do número de requisições atendidas para cada instância.

Tabela 2 - Mediana do número de requisições atendidas para cada instância do Retratos-PMAQ usando o *JMeter.*

Instância	Requisições Atendidas
Docker Standalone	4.003
OKD com 1 POD	4.542
OKD com 03 PODs	12.024

Fonte: Autoria própria (2020).

Observa-se que o "Retratos" em execução no OKD com 1 (um) POD conseguiu atender 13% (treze por cento) mais requisições que o *Docker* no formato *standalone*. Foram atendidas 4.542 requisições em 90 segundos pelo OKD contra 4003 requisições atendidas no *Docker standalone*.

Ao aumentar o número de PODs para 3 (três) no OKD, observa-se que a quantidade de requisições atendidas no decorrer dos testes foi 3 (três) vezes maior que no formato *Docker standalone*, pois, os PODs adicionais disparam novos containers que possibilitam usar os demais *cores* da máquina virtual que, no ambiente *standalone*, estavam ociosos. É importante ressaltar que o desempenho no tempo de resposta não depende unicamente do número de PODs definido no OKD, e sim dos recursos computacionais disponíveis no cluster onde a aplicação está sendo executada. Se um número elevado de PODs for definido e não houver recursos (CPU, memória, etc.) disponíveis, o cluster ficará congestionado e não haverá o crescimento linear no número de respostas com o aumento no número de requisições, além da degradação no tempo de resposta das requisições.

CONCLUSÃO

A infraestrutura proposta e testada possibilita ao desenvolvedor acompanhar o processo de *deploy* da aplicação, algo que era inviável no formato anterior (*Docker standalone*). No ambiente anterior, as atualizações de versões das aplicações ocorriam de forma automática através de agendamento na *crontab* do Linux. Portanto, um erro na definição da instalação de um pacote, ou algum bug que não foi identificado no processo de teste, poderia causar a indisponibilidade do sistema e, como o desenvolvedor não tem acesso aos *logs* de saída da aplicação para identificar e corrigir o problema, necessitaria do auxílio da equipe de infraestrutura para resolução de possíveis problemas.

No novo ambiente proposto, ao contrário do anterior, o desenvolvedor tem total controle sobre o processo de *deploy* da aplicação, acompanhando a execução dos

sistemas através do *dashboard* de gerenciamento do OKD.

Uma outra vantagem do novo ambiente é a possibilidade de escalar a infraestrutura do sistema para atender a um maior número de usuários de forma manual ou dinâmica, por exemplo, através de uma métrica como uso de CPU que poderia ser para disparar a criação de novos PODs para atender mais requisições. Para demonstrar as vantagens desta proposta, especificamente para o sistema, foram realizados testes de carga na infraestrutura legada, que utiliza o *Docker* em modo *standalone*, e em seguida, os mesmos testes foram realizados na nova infraestrutura proposta, que utiliza o OKD, primeiramente executando a aplicação com apenas 1 (um) POD e depois com 3 (três) PODs.

Os resultados obtidos demonstram que, mesmo com 1 (um) POD o OKD respondeu 13% mais requisições que o formato legado. Ao aumentar o número de PODs para 3 (três), os testes resultaram em 3 (três) vezes mais requisições respondidas e, com isso, demonstrando que a infraestrutura proposta é escalável e quando configurada em um cluster com mais recursos computacionais disponíveis, o número de PODs pode ser incrementado permitindo responder ainda mais requisições.

Por fim, é importante ressaltar que, embora a arquitetura apresentada possibilite escalar as aplicações e permita atender mais usuários, há pontos que este trabalho não abordou e que devem ser considerados em trabalhos futuros. Considerando a importância das aplicações que lidam com dados de saúde, como o Retratos da Atenção Primária à Saúde, sugerimos que mais estudos sejam realizados para propor o uso de uma plataforma de sistema de arquivos distribuídos que possibilita armazenar as mídias enviadas pelos usuários de forma segura e escalável, como o *Ceph* (CEPH, 2020) (UMRAO; HACKETT; SINGH, 2017). Um outro ponto a considerar é a possibilidade de uso de múltiplos links de internet para permitir uma maior disponibilidade do sistema.

No contexto de segurança da informação, um ponto a ser considerado é a inspeção em busca de vulnerabilidades nos pacotes utilizados pelos sistemas como, por exemplo, pacotes PIP do *Python*, de forma automática durante a execução do *pipeline* de CI/CD apresentado neste trabalho.

REFERÊNCIAS

BRASIL. Ministério da Saúde. **Portaria n° 1.645, de 2 de outubro de 2015**. Dispõe sobre o Programa Nacional de Melhoria do Acesso e da Qualidade da Atenção Básica (PMAQ-AB). Disponível em: http://bvsms.saude.gov.br/bvs/saudelegis/gm/2015/prt1645_01_10_2015.html. Acesso em: 2 fev. 2020.

CEPH. **Ceph webpage**. 2020. Disponível em: https://ceph.io. Acesso em: 2 fev. 2020.

DOCKER INC. **Docker**. Docker webpage, 2020. Disponível em: https://www.docker.com/resources/what-container. Acesso em: 3 fev. 2020.

DUMPLETON, G. **Deploying to Openshift**. [*S.l.*]: O'Reilly Media, 2018.

EBERT, C. *et al*. DevOps. **IEEE Software**, v. 33, n. 3, p. 94-100, 2016.

ERL, T.; PUTTINI, R.; MAHMOOD, Z. **Cloud Computing: Concepts, Technology & Architecture**. [*S.l.*]: Pearson Education, 2013.

JIANG, Y.; ADAMS, B. "Co-evolution of infrastructure and source code - An empirical study," **IEEE Int. Work. Conf. Min. Softw. Repos.**, v. 2015, p. 45–55, aug. 2015.

LOSSENT, A.; RODRIGUEZ PEON, A.; WAGNER, A. PaaS for web applications with OpenShift Origin. **J. Phys. Conf. Ser.**, v. 898, n. 8, p. 1–7, 2017.

MELL, P.; GRANCE, T. The NIST definition of Cloud Computing. **Computer Security**, set. 2011.

MINISTÉRIO DA SAÚDE – MS. **Programa Nacional de Melhoria do Acesso e da Qualidade da Atenção Básica (PMAQ-AB)**. Brasília: Ministério da Saúde, 2015.

OKD COMMUNITY. **CentOS install**: Github repository, 2019. Disponível em: https://github.com/okd-community-install/installcentos. Acesso em: 2 fev. 2020.

PAHL, C. Containerization and the PaaS Cloud. **IEEE Cloud Comput.**, v. 2, n. 3, p. 24–31, 2015.

REDHAT INC. "**RedHat Openshift**" OpenShift Website, 2020. Disponível em: https://www.openshift.com. Acesso em: 3 fev. 2020.

SAYFAN, G. **Mastering Kubernetes.** [*S.l.*]: Packt Publishing, 2018.

UMRAO, V.; HACKETT, M.; SINGH, K. **Ceph Cookbook.** [*S.l.*]: Packt Publishing, 2017.