COMO ATUAR COM CONTÊINERES NO AMBIENTE DO TRIBUNAL DE JUSTIÇA DO DISTRITO FEDERAL E TERRITÓRIOS:

USANDO APLICAÇÕES NO OPENSHIFT

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES
Instituto Brasileiro de Informação em Ciência e Tecnologia

PODER JUDICIÁRIO
Tribunal de Justiça do Distrito Federal e Territórios

COMO ATUAR COM CONTÊINERES NO AMBIENTE DO TRIBUNAL DE JUSTIÇA DO DISTRITO FEDERAL E TERRITÓRIOS:

Usando aplicações no OpenShift

Brasília
2021

PRESIDÊNCIA DA REPÚBLICA

Jair Messias Bolsonaro
Presidente da República

Hamilton Mourão
Vice-Presidente da República

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES

Marcos Cesar Pontes
Ministro da Ciência, Tecnologia e Inovações

INSTITUTO BRASILEIRO DE INFORMAÇÃO EM CIÊNCIA E TECNOLOGIA

Cecília Leite Oliveira
Diretora

Reginaldo de Araújo Silva
Coordenador de Administração (COADM)

Gustavo Saldanha
Coordenador de Ensino e Pesquisa, Ciência e Tecnologia da Informação (COEPE)

José Luis dos Santos Nascimento
Coordenado de Planejamento, Acompanhamento e Avaliação (COPAV)

Anderson Itaborahy
Coordenador-Geral de Pesquisa e Desenvolvimento de Novos Produtos (CGNP)

Bianca Amaro de Melo
Coordenadora-Geral de Pesquisa e Manutenção de Produtos Consolidados (CGPC)

Tiago Emmanuel Nunes Braga
Coordenador-Geral de Tecnologias de Informação e Informática (CGTI)

Milton Shintaku
Coordenador de Articulação, Geração e Aplicação de Tecnologia (COTEC)

PODER JUDICIÁRIO

TRIBUNAL DE JUSTIÇA DO DISTRITO FEDERAL E DOS TERRITÓRIOS

Des. Romeu Gonzaga Neiva
Presidente

Desa. Ana Maria Duarte Amarante Brito
1ª Vice-presidente

Desa. Sandra De Santis Mendes de Farias Mello
2ª Vice-presidente

Desa. Carmelita Indiano Americano do Brasil Dias
Corregedora

Camila Lucas Porto
Secretaria de Jurisprudência e Biblioteca - SEBI

Marcelo Hilario de Moraes
Subsecretaria de Biblioteca

Helen Barbosa
Serviço de Multimeios - SERMUT

Amanda Lopes de Araújo Soares
Subsecretaria de Doutrina e Jurisprudência - SUDJU

Marcelo Ribeiro da Silva
Núcleo de Revista Jurídica - NUREV

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES
Instituto Brasileiro de Informação em Ciência e Tecnologia

PODER JUDICIÁRIO
Tribunal de Justiça do Distrito Federal e Territórios

COMO ATUAR COM CONTÊINERES NO AMBIENTE DO TRIBUNAL DE JUSTIÇA DO DISTRITO FEDERAL E TERRITÓRIOS:

Usando aplicações no OpenShift

Rebeca dos Santos de Moura
Lucas Rodrigues Costa
Milton Shintaku

Brasília
2021

EQUIPE TÉCNICA
Diretora do Instituto Brasileiro de Informação em Ciência e Tecnologia
Cecília Leite Oliveira

Coordenador-Geral de Tecnologias de Informação e Informática (CGTI)
Tiago Emmanuel Nunes Braga

Coordenador do Projeto
Milton Shintaku

Autores
Rebeca dos Santos de Moura
Lucas Rodrigues Costa
Milton Shintaku

Design Gráfico, Diagramação e Ilustrações
Rafael Fernandez Gomes
Nuielle Medeiros

Normalização
Priscila Rodrigues dos Santos

Revisor
Rafael Teixeira de Souza
Flavia Karla Ribeiro Santos

Dados Internacionais de Catalogação-na-Publicação (CIP)

Bibliotecária: Ingrid Schiessl CRB1/ 3084

M929 Moura, Rebeca dos Santos

Como atuar com Contêineres no ambiente do Tribunal de Justiça do Distrito Federal e Territórios: Usando aplicações no OpenShift / Rebeca dos Santos de Moura; Lucas Rodrigues Costa; Milton Shintaku. -- Brasília: Ibict, 2021.

p. 36

ISBN 978-65-89167-17-4

DOI: 10.22477/9786589167174

1. Sistemas de informação. 2. Tecnologia da informação. 3. Software livre. I. Moura, Rebeca dos Santos. II. Costa, Lucas Rodrigues. III.Shintaku, Milton. IV. Título

CDU 681.3
CDD 600

Esta produção é um produto do Projeto de pesquisa Estudos para atualização tecnológica de ecossistema de informação do Tribunal de Justiça do Distrito Federal e Territórios.

Ref. IBICT - Processo SEI nº 01302.000390/2020-38

Ref. FUNDEP 28331

Setor de Autarquias Sul (SAUS) Quadra 05 Lote 06, Bloco H – 5º andar
Cep:70.070-912 – Brasília, DF -Telefones: 55 (61) 3217-6360/55 /(61)3217-6350 -www.ibict.br

SUMÁRIO

APRESENTAÇÃO

O presente Guia foi concebido no Projeto de Pesquisa firmado entre o Tribunal de Justiça do Distrito Federal e Territórios (TJDFT) e o Instituto Brasileiro de Informação em Ciência e Tecnologia (Ibict), voltado ao desenvolvimento de estudos para melhoria de alguns serviços informacionais do tribunal. Logo, contou com o apoio das equipes envolvidas no projeto, principalmente as ligadas à Coordenadoria-Geral de Tecnologia da Informação, que orientou a equipe do Ibict na preparação do ambiente utilizado para o desenvolvimento dos estudos.

O TJDFT utiliza dois *softwares* livres, o *Open Journal System (OJS)* e o *DSpace* na Revista de Doutrina Jurídica (RDJ) e na Biblioteca Digital, respectivamente, ou seja, dois sistemas informatizados não desenvolvidos pela equipe de informática do tribunal. Assim, se faz necessário atualizar tais softwares livres para o modelo utilizado atualmente no desenvolvimento e manutenção dos sistemas do tribunal. O novo modelo é baseado em contêineres e possui um processo diferenciado, com especificidades próprias e definições de boas práticas no desenvolvimento, gerenciamento e arquitetura dos sistemas. Visto que o modelo está completamente implementado, a equipe do Ibict atuará como se fosse manutenção dos sistemas.

Nesse sentido, este documento tem como objetivo apresentar a estrutura do ambiente computacional do TJDFT e exemplificar como devem ser atualizados os *softwares* livres para o modelo utilizado pelo tribunal, tendo como base a experiência da equipe do Ibict, que atua com

o *OJS* e o *DSpace*. Com isso, ajudam-se outros profissionais do TJDFT e de entidades que utilizam ambientes similares para atuar em situações semelhantes, mesmo que nem tudo esteja descrito com detalhes devido a questões de segurança.

1. INTRODUÇÃO

O TJDFT, alinhado a tendências no fomento à disseminação da informação, tem adotado tecnologias que disponibilizam informações no formato digital, relacionadas à sua atuação. Além disso, oferta, em seu portal, informações e serviços aos cidadãos, assim como serviços informacionais, por meio da biblioteca e editora. No que tange à documentação digital, o TJDFT disponibiliza vários canais, como a Biblioteca Digital e a Revista, implementados com os softwares livres *DSpace* e *OJS*, respectivamente

Da mesma forma, segue as tendências inovadoras na estruturação de seu ambiente computacional, visto que cada vez mais os órgãos de governo precisam da agilidade para suprir as necessidades de implementação, manutenção, atualização e evolução dos sistemas de informação. Essa estruturação da infraestrutura computacional tem evoluído desde o processo de *downsize*, no qual os computadores de grande porte foram gradualmente substituídos por vários servidores independentes e isolados.

Posteriormente, esses servidores foram integrados e particionados em máquinas virtuais, que simulavam a independência, mas que possibilitavam certas flexibilidades, principalmente na expansão de recursos ofertados. Esse passo foi importante para a formação de nuvens, que, nestes últimos anos, tem se firmado como tendência mundial na computação. Entretanto, a formação de nuvens ou *clusters* se aplica apenas à formação do ambiente em que os sistemas informatizados são hospedados.

Assim, para facilitar a instalação, atualização e manutenção dos sistemas foi criada a técnica de contêineres, desenvolvida pela Empresas *RedHat*, voltada

à doação de independência e flexibilidade aos ambientes dos sistemas. Os contêineres são baseados em micro serviços utilizados no gerenciamento de uma infinidade de aplicações, incluindo questões de armazenamento e segurança. Tanto que, na computação já existe o verbo "conteinizar", para o processo de empacotar aplicações fornecendo-lhes segurança, isolamento e outros serviços, facilitando o seu funcionamento e a sua portabilidade.

Nesse contexto, o TJDFT adota o uso de contêineres em suas aplicações, mesmo que alguns dos sistemas legados ainda não estejam neste ambiente, caso da Revista RDJ e da Biblioteca Digital do TJDFT (BDTJDFT). Ambos os sistemas foram implementados com *softwares* livres, *OJS* e *DSpace*, respectivamente, e por meio do projeto de pesquisa, foram colocados em contêiner.

2. AMBIENTE COMPUTACIONAL DO TJDFT

O ambiente computacional do TJDFT utiliza a abordagem de *DevOps*[1] (junção das palavras *development* e *operations*), que agrupa cultura, automação e design de plataforma com o objetivo de agregar mais valor aos negócios e aumentar sua capacidade de resposta às mudanças por meio de entregas de serviços rápidas e de alta qualidade.

A cultura *DevOps* auxilia instituições no gerenciamento de lançamento de novas versões dos softwares, estimulando a comunicação entre as equipes de Desenvolvimento e Operações. Dessa forma, eventos podem ser acompanhados com maior facilidade, assim como o controle de processos documentados e a emissão de relatórios granulares. Instituições com problemas no processo de implantação de novas versões até possuem automação, mas querem maior flexibilidade para gerenciar e conduzir esse processo, sem precisar editar tudo na linha de comando. Idealmente, essa automação deve ser disparada por recursos não operacionais, em ambientes específicos, que não estejam na produção. Assim, o desenvolvedor ganha maior controle sobre o ambiente de desenvolvimento e produção, e o administrador da infraestrutura, um maior entendimento sobre os aplicativos e cenários.

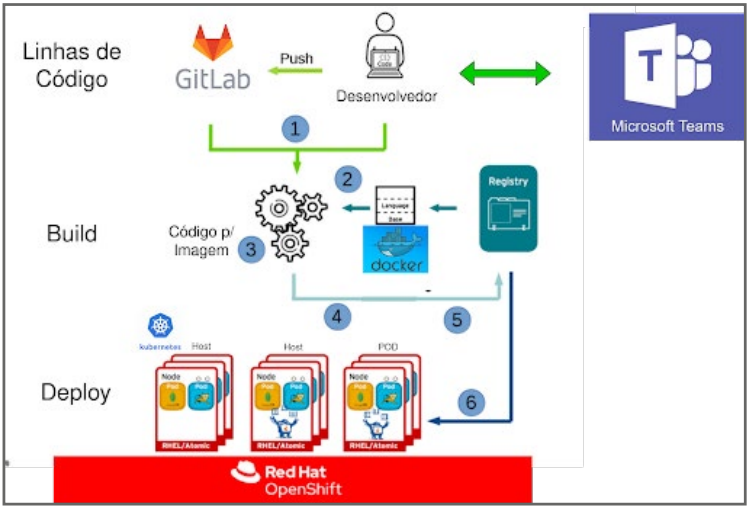
Nesse contexto, um modelo de *DevOps* encurta o ciclo de vida de desenvolvimento de sistemas e fornece a entrega contínua com alta qualidade de software. Esse modelo facilita a integração entre os times de desenvolvimento

1 Disponível em: https://www.redhat.com/pt-br/topics/devops

e infraestrutura, a gerência de ambientes e sistemas, além de promover maior aderência ao ambiente de execução das aplicações desde o início do desenvolvimento.

Existem várias ferramentas de código aberto ou disponíveis comercialmente para implementação do modelo *DevOps* (Figura 1). O TJDFT utiliza, primariamente, duas delas: *OpenShift* e *GitLab*, que, por sua vez, empregam *Kubernetes* e *Docker*. Além disso, a plataforma *Microsoft Teams* é usada para a comunicação entre as equipes envolvidas no projeto.

Figura 1 - Estrutura de Infraestrutura do modelo DevOps do TJDFT.



Fonte: Os Autores (2021).

A Figura 1 exemplifica os relacionamentos entre as ferramentas utilizadas nesse modelo *DevOps* em seis passos:

1. O desenvolvedor realiza um *push* no *GitLab*;
2. O *pipeline* de CI/CD é ativado;

3. A leitura do *DockerFile* definido no código é realizada;
4. A imagem *Docker* é construída;
5. A imagem é enviada para o *Registry* (repositório de imagens *Docker*);
6. O *OpenShift* busca a imagem no *Registry* e usa o *Kubernetes* para criar os *hosts*.

Por fim, toda interação entre as equipes é realizada no *Microsoft Teams*, em todos os passos.

2.1 GITLAB

O *GitLab*[2] é um gerenciador de repositório de *software* baseado em *git*, que permite o versionamento de código, com suporte a *Wiki*, gerenciamento de tarefas e *Continuous Integration* (CI)/*Continuous Delivery* (CD), muito comumente chamadas apenas de CI/CD.

A CI/CD se define como integração contínua e entrega contínua. A integração contínua facilita e agiliza o teste de novas funções em um ambiente semelhante ao de produção. O desenvolvedor só precisa realizar o *deploy* depois que todos os testes (realizados no próprio *GitLab*) passarem, o que resulta na qualidade e velocidade de entrega de *softwares* maiores no ambiente.

O caminho pelo qual o *software* passa ao longo do CI/CD é chamado *pipeline*, que é executado no *Runner* do *GitLab*. O *Runner* é a máquina onde o CI/CD será rodado. Ele pode estar localizado em alguma nuvem, como a *Amazon Web Services* (AWS), algum servidor ou até mesmo localmente, no computador do desenvolvedor. Um *pipeline* de integração contínua pode ser feito de diversas formas, dependendo muito das ferramentas de que o projeto necessita.

2 Disponível em: https://docs.gitlab.com/ee/ci/

No TJDFT, o *OpenShift* não faz o *build* (geração do código executável das aplicações) de projetos por questões de segurança, repassando o trabalho à ferramenta de entrega contínua do *GitLab* a partir do código dos projetos mantidos na plataforma. Esse sistema utiliza o *Docker* para isso.

2.2 DOCKER

Docker[3] é uma plataforma *open source* que facilita a criação e administração de ambientes isolados. Com ele, é possível empacotar uma aplicação ou um ambiente dentro de um contêiner, como se fosse uma máquina virtual modular e extremamente leve. O *Docker* trabalha à maneira de um conjunto de produtos de plataforma como serviço (PaaS), realizando a virtualização no nível do sistema operacional e entregando *softwares* em pacotes denominados de contêineres. Os contêineres são isolados uns dos outros e agrupam seus próprios *softwares*, bibliotecas e arquivos de configuração. Eles podem se comunicar uns com os outros por meio de canais bem definidos. Todos os contêineres são executados por um único kernel do sistema operacional e, portanto, usam menos recursos do que as máquinas virtuais.

A virtualização do kernel da máquina hospedeira (*host*) é compartilhada com a máquina virtual ou o *software*. Dessa forma, um desenvolvedor pode agregar ao seu contêiner todas as bibliotecas e outras dependências do seu sistema. O *docker* torna as operações em uma infraestrutura como serviços web mais intercambiáveis, eficientes e flexíveis.

Em conjunto com o *Kubernetes*, o *Docker* ainda realiza as mesmas tarefas do seu objetivo original, com a diferença de que o sistema automatizado solicita que o Docker realize essas tarefas, em vez de o administrador fazer as solicitações manualmente.

3 Disponível em: https://www.redhat.com/pt-br/topics/containers/what-is-docker

2.3 KUBERNETES

O *Kubernetes*,[4] por sua vez, é uma plataforma *open source* que automatiza a implantação, além de dimensionar e gerenciar os aplicativos em contêineres. Um dos objetivos da criação do *Kubernetes* e do gerenciamento de contêineres é levar mais eficiência ao time de desenvolvimento, uma vez que o transporte de aplicações inteiras é complexo e improdutivo. Uma opção, em vista de solucionar este problema, é dividi-la em diversos contêineres, com os códigos e recursos encapsulados.

Essa plataforma possibilita a criação e o gerenciamento de um *cluster* de contêineres em nuvens privadas ou públicas, eliminando grande parte dos processos manuais necessários à implantação e escalação das aplicações em contêineres. Dessa forma, o gerenciamento dos *clusters* é feito com facilidade e efetividade, garantindo que todos estejam em perfeito funcionamento e dimensionando todos os contêineres, caso algum deles fique inativo.

2.4 OPENSHIFT

O *OpenShift*, desenvolvido pela *Red Hat*[5], é uma plataforma de contêineres *Kubernetes* para empresas com operações automatizadas em todo o conjunto de sistemas. Com ele, é possível criar, desenvolver e implantar aplicações de forma simples e rápida, em qualquer infraestrutura. A plataforma inclui um sistema operacional Linux, ambiente de execução de contêiner, rede, monitoramento, registro e soluções de autorização e autenticação. Assim, a integração de arquitetura, processos, plataformas e serviços necessários à impulsão do trabalho das equipes de desenvolvimento e de operações é feita de forma aliada e harmoniosa.

4 Disponível em: https://www.redhat.com/pt-br/topics/containers/what-is-kubernetes
5 Disponível em: https://www.redhat.com/pt-br/technologies/cloud-computing/openshift

O *Openshift* trouxe um novo tipo de virtualização, chamado contêiner, muito eficiente para o abastecimento de ambientes aos desenvolvedores de aplicações, contribuindo para maior produtividade, autonomia e velocidade de publicação de sistemas.

2.5 MICROSOFT TEAMS

O *Microsoft Teams*[6] é uma plataforma unificada de comunicação e colaboração que proporciona espaços para *chat* (bate-papo), videoconferências, armazenamento de arquivos e integração de aplicativos no local de trabalho. Apresentado como ferramenta do *Office 365*, ele foi desenvolvido para facilitar a conversa, a participação e o compartilhamento de informações, além de promover a cooperação entre pessoas e equipes dentro de uma empresa.

O TJDFT utiliza as ferramentas de *DevOps* para automatizar a entrega de suas aplicações com testes de qualidade, desenvolvimento de recursos e lançamentos de manutenção, com vistas a incrementar a confiança, a segurança e o desenvolvimento rápido, seguindo ciclos. Por fim, a ferramenta de comunicação é vital para o conhecimento e acompanhamento de todos os membros da equipe, incluindo pessoas fora da área técnica de tecnologia da informação, que serão eventuais usuários das aplicações desenvolvidas.

6 Disponível em: https://www.microsoft.com/pt-br/microsoft-teams

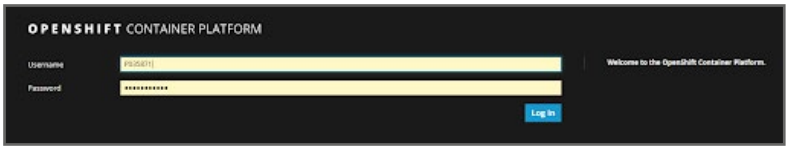
3. PROCESSO DE LEVAR APLICAÇÕES PARA OPENSHIFT

Nesta seção apresentaremos um guia de uso para as várias ferramentas do TJDFT. Além da configuração do *Docker*, são necessários acessos aos ambientes internos do TJDFT do *OpenShift*[7] e *GitLab*[8]. O regimento do TJDFT solicita o preenchimento de um formulário de solicitação de acesso para a rede interna. Após o envio e processamento da solicitação, são fornecidos login e senha para acesso à VPN e demais sistemas internos.

3.1 ACESSO AO PROJETO NO OPENSHIFT

A partir do acesso à rede interna (conexão VPN), é necessário logar no OpenShift, disponível em https://os311.tjdft.jus.br/, como mostra a Figura 2.

Figura 2 - Login no OpenShift.

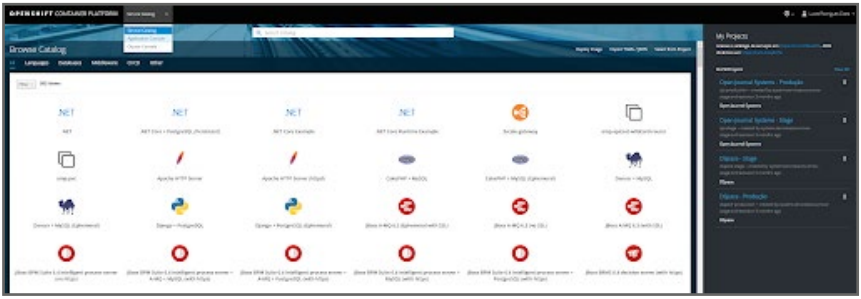


Fonte: Captura de tela (2021).

7 Disponível em: https://os311-console.tjdft.jus.br/
8 Disponível em: https://gitlab.tjdft.jus.br/

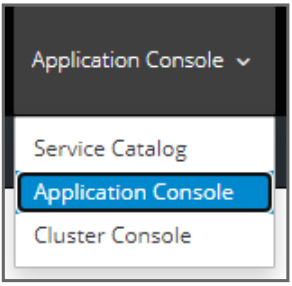
Após o Login, o *OpenShift* exibe o painel de controle do sistema, mostrado na Figura 3. Deve-se entrar no *Application Console*, apresentado na Figura 4.

Figura 3 - Painel de controle do OpenShift.



Fonte: Captura de tela (2021).

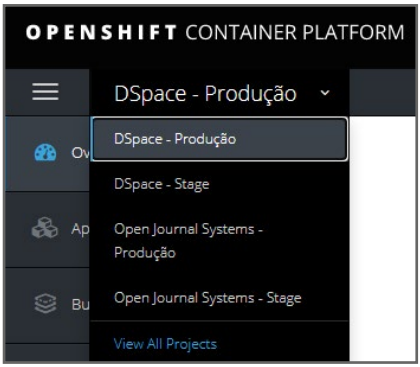
Figura 4 - Selecionar Application Console.



Fonte: Captura de tela (2021).

Independente do sistema em questão, os projetos são desenvolvidos em dois ambientes no *OpenShift*: *Production e Stage*. O primeiro se refere a projetos no ambiente de Produção e o segundo, no ambiente de homologação. Essa distinção é para melhor organização dentro da ferramenta. Para gerenciamento do acesso dos colaboradores aos projetos, é necessário escolher o projeto (também chamado de aplicação) desejado, como mostra a Figura 5.

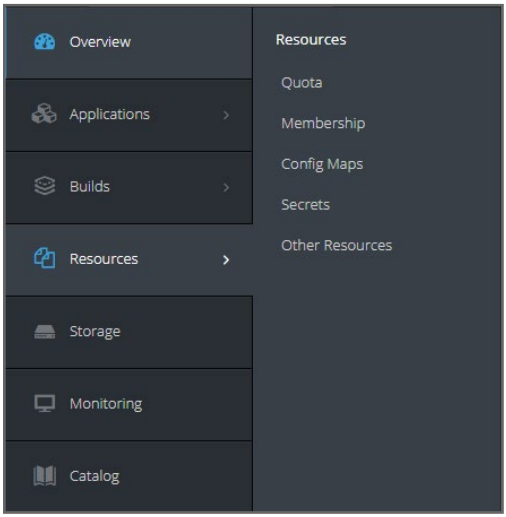
Figura 5 - Escolha do projeto.



Fonte: Captura de tela (2021).

Após a seleção do projeto (seja *Production* ou *Stage*), deve-se entrar em ***Resources*** e selecionar ***Membership***, conforme Figura 6.

Figura 6 - Recursos do projeto.



Fonte: Captura de tela (2021).

Na página de ***Membership***, deve-se clicar em "***Edit Membership***", exibido na Figura 7.

Figura 7 - Editando o Membership.

Fonte: Captura de tela (2021).

Na linha que aparece para inserção, é necessário clicar em "***Show hidden roles***", apresentado na Figura 8, para mostrar os perfis de usuários adicionais.

Figura 8 - Apresentação dos papéis.

Fonte: Captura de tela (2021).

A Figura 9 mostra os perfis de usuários disponíveis para os projetos *OpenShift*. A seguir, apresentamos a definição dos perfis mais usados:

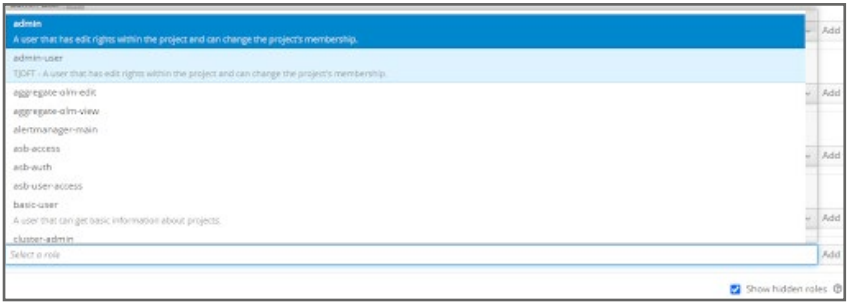
- *admin-user*: acesso à maior parte dos objetos dentro do projeto, com as mesmas permissões do perfil *edit* e a função de acrescentar permissão de criar role, *rolebinding*, e dar/retirar permissão de outros membros dentro do projeto;

- *edit*: acesso a objetos restritos dentro do projeto, faz *build*, exclui *pod*, escala *pod*, altera *secrets*, *configmap*, *imagestream*, entre outros;

- *view*: acesso a objetos restritos dentro do projeto, conseguindo apenas visualizá-los.

Para os demais perfis, é necessário solicitar informações ao SERPLA.

Figura 9 - Perfis disponíveis no projeto.



Fonte: Captura de tela (2021).

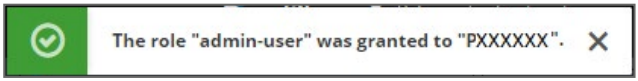
Neste projeto, será usado o perfil *admin-user* para todos os participantes. Assim, após escolher o perfil na lista suspensa da Figura 10, deve-se inserir o ponto no formato "P" + os seis primeiros dígitos do CPF (exemplo P123456 ou PXXXXXX) do participante em questão e clicar em "***Add***".

Figura 10 - Inserção do ponto do usuário para seleção do perfil.

Fonte: Captura de tela (2021).

Após o clique, uma confirmação é exibida na parte superior direita da tela, indicando que o perfil escolhido foi atribuído ao participante em questão, mostrado na Figura 11.

Figura 11 - Mensagem de confirmação



Fonte: Captura de tela (2021).

Agora, é necessário repetir os passos em cada um dos projetos (tanto *Production* como *Stage*), para cada membro da equipe que precisa da permissão.

3.2 DOCKER

O primeiro passo para a conteinerização de um ambiente é a criação do *Dockerfile*. Assim, é necessário instalar o Docker para criar a primeira imagem do projeto. Uma forma rápida e fácil de instalar a versão mais atual da ferramenta é executar o seguinte comando:

```
$ curl -fsSL https://get.docker.com | sh
```

A imagem *docker* é representada por um arquivo de configuração chamado *Dockerfile*. Neste arquivo são definidas as especificações da imagem, isto é, como ela deverá ser construída, e os comandos necessários à sua correta configuração. Um exemplo de *dockerfile* é mostrado a seguir.

```
FROM http:2.4.46-alpine
COPY var/www/hmtl/ /var/www/hmtl/
CMD ["/bin/sh"]
```

Resumidamente, este *Dockerfile* indica que a imagem *Docker* é baseada na imagem http:2.4.46-alpine, disponível no *Docker Hub* (um repositório público para imagens *Docker*). Em seguida, é feita a cópia da pasta local ***var/www/hmtl/*** para dentro do contêiner. Por fim, o comando de execução ***/bin/sh*** é enviado.

Alguns dos principais comandos da ferramenta *Docker* são:

$ docker --version - ver a versão do *docker*;

$ docker ps -a - mostrar todos os processos *dockers* executando ou já executados;
$ docker build -t <image name>:1.0 - criar uma imagem;
$ docker run -ti <image name> /bin/bash - rodar o bash dentro do contêiner. As opções são ***t***, para terminal, e ***i***, para para interação.

3.3 GITLAB

No *GitLab* é criado o repositório de versionamento de códigos onde o projeto será mantido. Nesse repositório são acrescentados o *Dockerfile* (exemplificado na seção anterior) e o arquivo de configuração do *pipeline* de CI, chamado "*gitlab-ci.yml*".

Um exemplo desse arquivo é apresentado abaixo.

```
variables:
  IMAGE_NAME: ojs-stage/ibict-ojs
  IMAGE_NAME_PROD: ojs-production/ibict-ojs
stages:
    - deploy

deploy_desenv:
  stage: deploy
  image: docker:latest
  only:
    - master
  variables:
    IMAGE_TAG: desenv
script:
  - docker login -u $SA_USER -p $SA_USER_TOKEN_DESENV $REGISTRY_ADDR
  - docker build --pull -t $REGISTRY_ADDR/$IMAGE_NAME:$IMAGE_TAG .
  - docker push $REGISTRY_ADDR/$IMAGE_NAME:$IMAGE_TAG
```

```
deploy_prod:
  stage: deploy
  image: docker:latest
  only:
    - tags
  variables:
    IMAGE_TAG: prod
  script:
    - docker login -u $SA_USER -p $SA_USER_TOKEN_PROD $REGISTRY_ADDR
    - docker build --pull -t $REGISTRY_ADDR/$IMAGE_NAME_PROD:$IMAGE_TAG.
    - docker push $REGISTRY_ADDR/$IMAGE_NAME_PROD:$IMAGE_TAG
```

Basicamente o arquivo "*gitlab-ci.yml*" descreve um conjunto de ações (job) que são executadas para enviar as imagens definidas no *Dockerfile* para o *Registry* do *Openshift*.

As variáveis são dados utilizados durante a execução do *script* e podem ser definidas no contexto global ou local em cada *job*.

Os estágios (*stages*) definem o tipo de *job* que será executado. Em geral, são definidos estágios de *deploy* e *build*.

Em seguida, são definidos os nomes dos *jobs* no arquivo. No exemplo apresentado, temos "*deploy_desenv*" e "*deploy_prod*". Cada *job* é ligado a um *stage* e uma imagem *Docker* (*image*), e possui uma *flag* de *only*, que controla quando o *job* será executado. Por exemplo: o "*only: - master*", significa que todo *commit* feito na *branch master* do repositório emite uma ordem de execução para este *job*. Em outras palavras, essa configuração associa o *commit* ao *job*. No exemplo apresentado, o *job* "*deploy_prod*" só será executado quando uma *tag* for criada no repositório.

Por fim, os comandos de cada *job* que enviam as imagens para o *Registry* são definidos na *flag* de *script* por meio de comandos *Docker*.

Além disso, são necessárias algumas variáveis de ambiente no projeto do *GitLab* para o correto funcionamento dos *jobs* do arquivo . Neste exemplo, o arquivo utiliza as seguintes variáveis:

$SA_USER - usuário para login do *Registry*;
$SA_USER_TOKEN_DESENV - senha para login do *Registry* de desenvolvimento;
$SA_USER_TOKEN_PROD - senha para login do *Registry* de produção;
$REGISTRY_ADDR - endereço do *Registry*.

A equipe de TI do TJDFT deve configurar essas variáveis de ambiente e avisar aos desenvolvedores.

3.4 OPENSHIFT

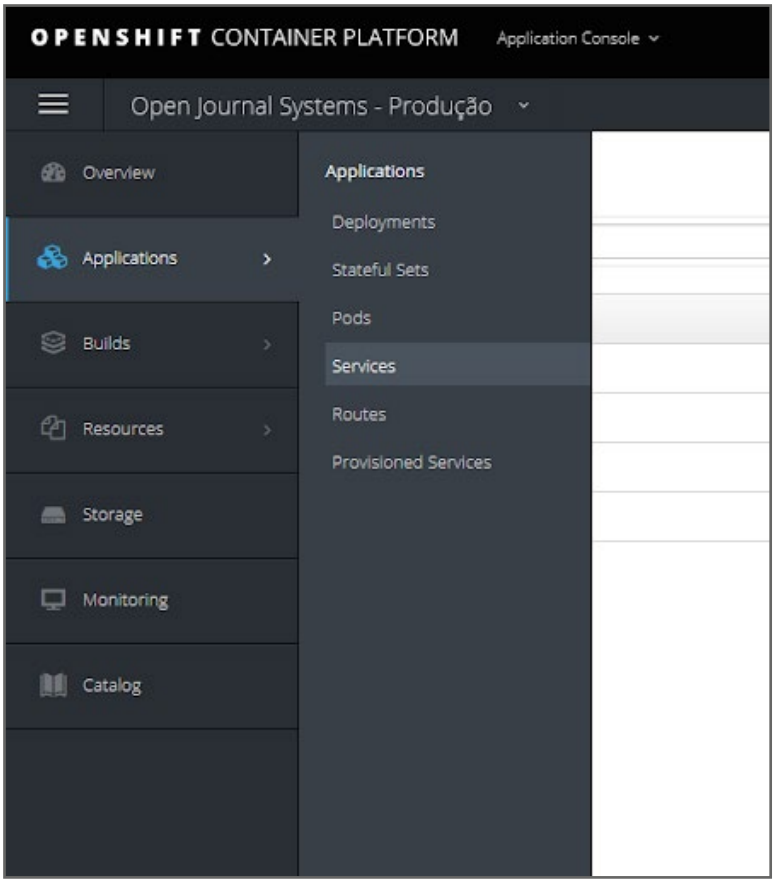
O *OpensShift* é um orquestrador dos contêineres *Kubernetes* em *Docker*, que são construídos a partir das imagens enviadas pelos *jobs* ao *Registry*. Nesta seção, será apresentado como acessar o *OpenShift* pelo navegador web e pela Interface de Linha de Comando (CLI). Alguns comandos básicos serão apresentados, bem como algumas definições dos elementos que compõem o *OpenShift*.

3.4.1 Acesso pelo navegador

Acessando o *OpenShift* pelo navegador[9], é possível acessar os diversos recursos e elementos pelo menu lateral "*Application*", mostrado na Figura 12.

9 Disponível em: https://os311.tjdft.jus.br/

Figura 12 - Serviços no OpenShift.



Fonte: Captura de tela (2021).

As seções a seguir fornecem definições sobre os principais conceitos e elementos que são encontrados ao usar o *OpenShift*. Muitos desses elementos vêm do *Kubernetes*, que é estendido pelo *OpenShift* para fornecer uma plataforma de ciclo de vida de desenvolvimento mais rica em recursos. As

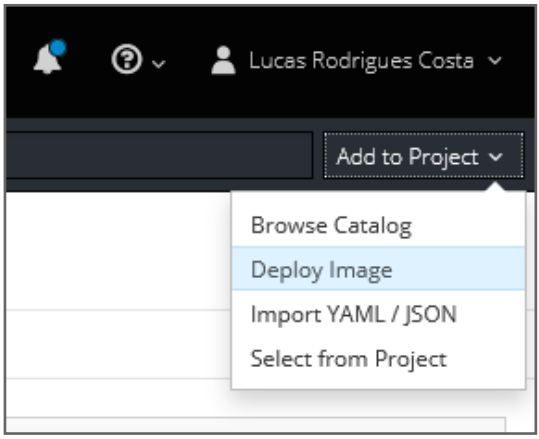
informações detalhadas podem ser encontrados no manual de referência on-line do próprio *OpenShift*[10].

3.4.1.1 Deployment

A ação de *deployment* efetivamente implanta uma imagem de contêiner que se encontra no *Registry*, passando a se chamar "aplicação". A aplicação representa um tipo de agrupamento acessível para determinados recursos de uma carga de trabalho.

Para realizar o *deploy* de uma aplicação, deve-se ir a ***Applications*** > ***Deployments*** no menu mostrado na Figura 13. Na página ***deployments***, deve-se clicar no menu superior direito "***Add to Project***" e selecionar a opção "***Deploy Image***", conforme Figura 13.

Figura 13 - Realizando um Deploy.



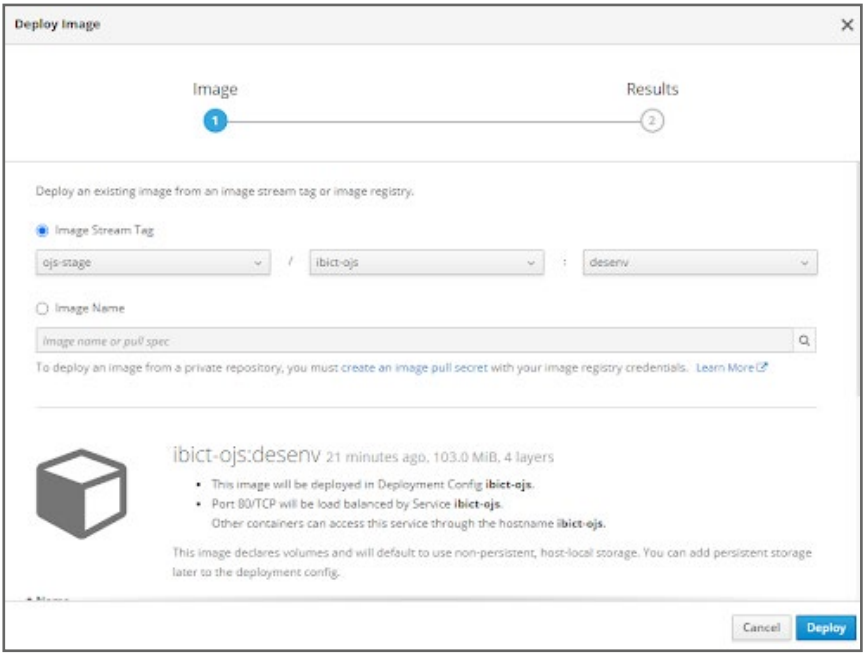
Fonte: Captura de tela (2021).

10 Disponível em: https://docs.openshift.com/enterprise/3.0/architecture/core_concepts/pods_and_services.html

Será aberta uma janela, mostrada na Figura 14, na qual é possível acrescentar uma imagem para o *deploy* de duas maneiras: *Image Stream Tag* e *Image Name*.

A primeira opção (*Image Stream Tag*) é usada quando a imagem é importada anteriormente para o *Registry* do *OpenShift* usando os *jobs* do *GitLab*. A segunda opção (*Image Name*) permite selecionar imagens que residem em um registro de imagem externo. Pode ser o *Docker Hub Registry*[11] ou qualquer outro registro de imagem acessível.

Figura 14 - Escolhendo uma imagem para o deploy.



Fonte: Captura de tela (2021).

11 Disponível em: https://hub.docker.com/

Após a escolha do projeto e da imagem, com sua respectiva tag, adicionalmente pode-se inserir variáveis de ambiente ao contêiner e modificar seu nome. Ao clicar no botão "Deploy" a aplicação será criada juntamente com seu Pod e seus Serviços, explicados a seguir.

3.4.1.2 Pods

O *Pod* dentro do *OpenShift* é a menor unidade de computação que pode ser definida, implantada e gerenciada, e representa um ou mais contêineres implantados juntos em um *host*.

Cada *Pod* é alocado com seu próprio endereço IP interno e uma porta. Os contêineres dentro dos *pods* podem compartilhar os recursos de armazenamento e rede.

O *OpenShift* trata os *pods* como amplamente imutáveis, de forma que alterações estruturais não podem ser feitas enquanto um *pod* está em execução. As mudanças são implementadas encerrando um *pod* existente e recriando-o com as configurações modificadas.

Os *pods* são tratados como dispensáveis e não mantêm o estado quando recriados. Portanto, devem ser gerenciados pelos códigos versionados no *GitLab*.

3.4.1.3 Serviços

Um serviço (*service*) atua como um balanceador de carga interno para os *pods*. Ele é associado a um grupo de *pods* e designa para eles um IP e *hostname*. O *service* realiza o balanceamento das requisições para todos os *pods* em execução.

Por exemplo, quando um *pod* que contém a camada de *frontend* da aplicação deseja se integrar a outro *pod* que contém o banco de dados, o primeiro *pod* realiza a chamada ao *service*, que envia a requisição para o segundo *pod*.

Em geral, um *service* é criado automaticamente, quando é realizado o *deploy* de uma aplicação que possui um serviço de rede.

3.4.1.4 Rotas

Uma rota (*route*) permite que os *pods* (ou conjunto de *pods*) sejam acessados externamente ao *cluster* do *OpenShift*. Ela redireciona a requisição para o *service*, que, posteriormente, roteia o tráfego aos *pods* associados a este *service*.

A equipe de TI do TJDFT disponibiliza um guia para a criação correta das rotas no *OpenShift*[12]. As instruções orientam criar uma rota segura para a aplicação. Uma rota segura usa TLS/HTTPS. Uma rota é uma referência ao serviço do *pod*, que deve expor uma determinada porta.

Nas configurações do *OpenShift* os tipos de *Termination* de uma rota, atualmente, são: *Edge*, *Reencrypt* ou *Passthrough*. A recomendação é usar a terminação *Edge*.

Uma rota segura é publicada com HTTPS (porta 443) ou possui redirecionamento das requisições de HTTP (porta 80) para HTTPS (porta 443), como mostra o vídeo no endereço acessado na rede interna[13].

Configurações adicionais são necessárias para expor a rota para fora da rede do TJDFT. Dessa forma, após a criação da rota, deve-se usar a interface

12 Disponível em: https://gitlab.tjdft.jus.br/infraestrutura/openshift/integracao/dns
13 Disponível em: https://drive.tjdft.jus.br/index.php/s/D7dYX65pNi3ggAs

CLI para expor a rota para a Internet. Inicialmente, é necessário identificar o nome da rota, sendo que o seguinte comando é usado:

```
$ oc get route
```

Com o identificador (id) da rota em mãos, deve-se expor a rota com o comando:

```
$oc label route <id_da_rota> -n <nome_do_projeto> “environment=internet”
```

Com isso, a aplicação estará disponível no ambiente de produção externo e acessível aos usuários finais.

3.4.2 Acesso pelo terminal

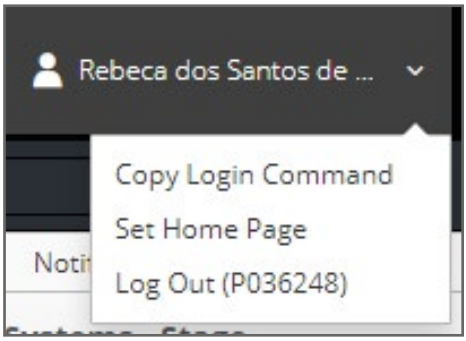
O *OpenShift* dispõe de uma interface de linha de comando (CLI) que permite criar aplicativos e gerenciar projetos a partir de um terminal. A CLI está disponível ao usar o comando oc:

```
$ oc <comando>
```

A equipe de TI do TJDFT disponibilizou os executáveis para instalação e execução da CLI em ambiente Linux, Windows e MacOS, assim como um manual de instruções.

Para acessar o *OpenShift* pelo terminal é necessário ter o oc instalado e estar conectado à VPN do TJDFT. Em seguida, deve-se obter o token de acesso à API a partir da aplicação do *OpenShift*. Uma forma de obter o token é buscando no menu de usuário logado na interface web a opção “*Copy Login Command*”, conforme Figura 15.

Figura 15 - Cópia do comando do login.



Fonte: Captura de tela (2021).

Essa opção copia, para a área de transferência do computador, a seguinte linha de comando(por exemplo):

```
$ oc login https://os311.tjdft.jus.br:443 --token=XXXXX
```

Ao executar o comando no terminal, o usuário realizará o login na CLI. Um exemplo de resposta bem sucedida é o seguinte:

```
Logged into "https://os311.tjdft.jus.br:443" as "P036248" using the token
provided.

You have access to the following projects and can switch between them with
'oc project <projectname>':

  * <project1_name>
    <project2_name>
    <project3_name>
    <project4_name>
```

```
Using project "dspace-production".
Welcome! See 'oc help' to get started
```

São exibidos os projetos aos quais o usuário tem acesso. É possível escolher qual projeto interagir com seguinte comando:

```
$ oc project <project_name>
Now using project "<project_name>" on server "https://os311.tjdft.jus.br:443".
```

Para exibir os pods existentes no projeto, usa-se o seguinte comando:

```
$ oc get pods
NAME                    READY    STATUS    RESTARTS   AGE
ibict-dspace-test-21-g8chx  1/1      Running   0        13h
postgres-ibict-3-m9k26      1/1      Running   0        7d
```

Conhecendo o nome do pod, é possível acessar seu terminal ou transferir arquivos entre a máquina hospedeira e o pod com os seguintes comandos.

- Para abrir um terminal remoto no contêiner:

```
$ oc rsh <pod_name>
```

- Para copiar o conteúdo de um diretório local para um diretório em um *pod*:

```
$ oc rsync <local_dir> <pod_name>:<pod_dir>
```

- Para copiar o conteúdo de um diretório em um *pod* para um diretório local:

```
$ oc rsync <pod_name>:<pod_dir> <local_dir>
```

Existem outros comandos que podem ser executados na interface de linha de comando. Esses comandos podem ser encontrados no manual de referência *on-line* do próprio *OpenShift*[14].

14 Disponível em: https://docs.openshift.com/container-platform/3.11/cli_reference/index.html

4. CONSIDERAÇÕES FINAIS

O presente guia foi criado especialmente para o ambiente computacional do TJDFT, conforme os padrões utilizados pela equipe de infraestrutura do tribunal. Entretanto, segue as orientações padronizadas do uso de contêiner. Sendo assim, pode ser útil a outras instituições que estão utilizando contêineres para manter os seus sistemas, ou estão em processo de adoção dessa metodologia.

Por isso, neste guia foram apresentadas as ferramentas utilizadas no modelo DevOps, usado no ambiente computacional da infraestrutura do TJDFT. Em seguida, foi descrito o processo de levar uma aplicação para contêiner que será gerenciado no *OpenShift*. Assim, apresenta toda a cadeia utilizada na gestão de sistemas informatizados em contêineres, apresentando todos os passos necessários, principalmente no uso de *softwares* livres.

Cabe destacar que, diante da imensidade de ofertas, cada vez mais ferramentas livres têm sido utilizadas por instituições e órgãos de governo , muitos mantidos por instituições tradicionais de ensino e pesquisa. Entretanto, em grande parte dos casos, a documentação ainda é um problema, visto que muitas vezes só existe documentação em inglês ou fragmentada em várias obras.

Em vista disso, este guia procurou atender a todos os passos necessários para colocar uma aplicação, principalmente as de software livres, em contêiner, segundo as orientações seguidas pelo TJDFT. Procurou contribuir para a documentação do projeto e, da mesma forma, atender a equipe de infraestrutura do tribunal, para com isso, atender à necessidade de documentação técnica em português sobre o uso de contêiner.

ISBN: 978-65-89167-28-0

TJDFT
PODER JUDICIÁRIO DA UNIÃO
TRIBUNAL DE JUSTIÇA DO
DISTRITO FEDERAL E DOS TERRITÓRIOS

ibict
Instituto Brasileiro de Informação em Ciência e Tecnologia
UNIDADE DE PESQUISA DO MCTI

MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÕES

PÁTRIA AMADA
BRASIL
GOVERNO FEDERAL