# Guia para publicação de aplicações na Cloud Privada

# 1 - Introdução

Este guia contém as orientações básicas sobre o processo de Build e Deploy de aplicações para o ambiente de Cloud interna da empresa.

# 2 - Visão geral do processo

O processo para levar a aplicação desde o seu código fonte inicial até a sua execução final no ambiente *Kubernetes* requer um conjunto de etapas que precisam ser cumpridas.

Fazendo um mapeando dos principais pontos desse processo podemos dividi-lo da seguinte maneira:

Tomando como base as 5 etapas acima, segue abaixo um detalhamento de cada uma delas.

# 3 - Etapas

## 3.1 – Dockerfile

O Dockerfile é um arquivo texto que descreve as etapas e instruções que o Docker precisa para preparar uma imagem, incluindo a instalação de pacotes, criação de diretórios e definição de variáveis de ambiente entre outras coisas.  
Para mais detalhes sobre o assunto, seguem abaixo algumas referências:

* <https://www.mundodocker.com.br/o-que-e-dockerfile/>
* <https://docs.docker.com/get-started/02_our_app/#build-the-apps-container-image>

Para auxiliar no desenvolvimento do Dockerfile, foram selecionados alguns arquivos de exemplo dentro das principais tecnologias de desenvolvimento da empresa: Java, .Net, Angular e PHP. Estes arquivos podem ser acessados neste repositório: <https://github.com/AldoCarlos/pa.templates>

Os arquivos foram construídos com base nas Aplicações Modelo (APM’s) de cada tecnologia e também baseados em projetos que já foram publicados no ambiente de Cloud da empresa. A ideia é que os arquivos possam servir como uma referência técnica inicial para a confecção do Dockerfile da aplicação.

Para realizar testes durante o desenvolvimento local do arquivo, é necessário ter instalado a ferramenta do Docker. Ver o post [Docker Desktop deixa de ser gratuito](http://desenvolvimento.blog.intra.rs.gov.br/2022/01/27/docker-desktop-deixa-de-ser-gratis/) para obter mais detalhes sobre a instalação.

Tendo o Docker instalado e o arquivo Dockerfile no diretório raiz da aplicação, basicamente, serão necessários apenas 2 comandos para montar e rodar a aplicação:   
  
> docker build -t apm-angular:latest .

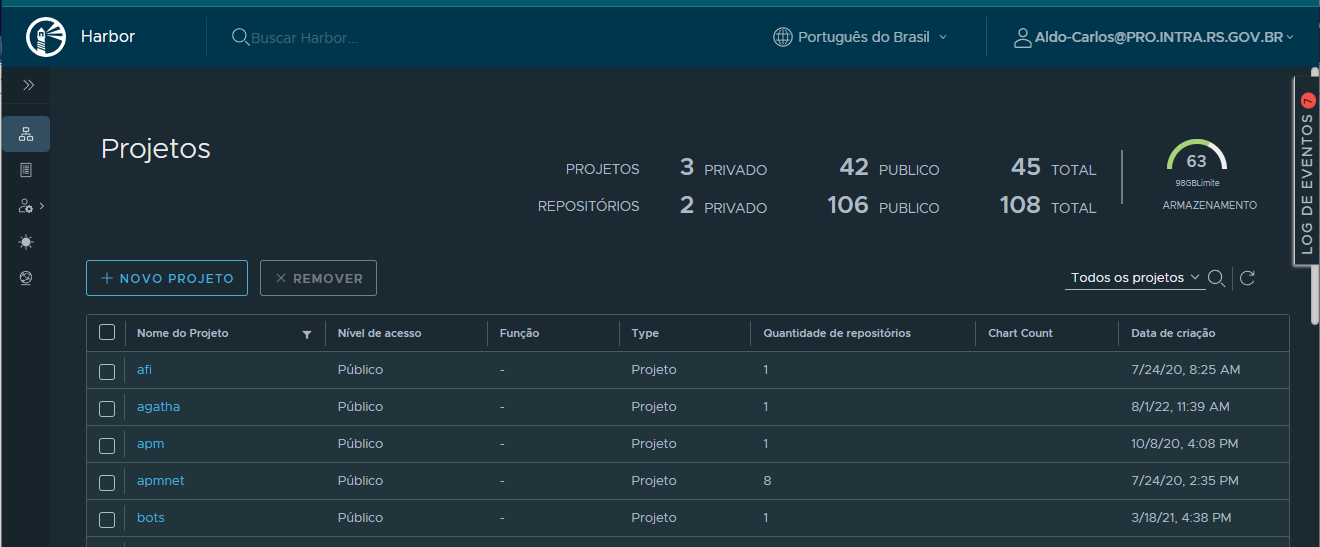
> docker run -d -p 8080:80 apm-angular:latest

A referência completa para os comandos do Docker CLI [pode ser consultada aqui](https://docs.docker.com/engine/reference/commandline/cli/).

## 3.2 – Harbor

O Harbor é o repositório oficial de imagens Docker da empresa. Também chamado de *Registry*, o Harbor pode ser acessado no endereço: <https://harbor001.pro.intra.rs.gov.br/>

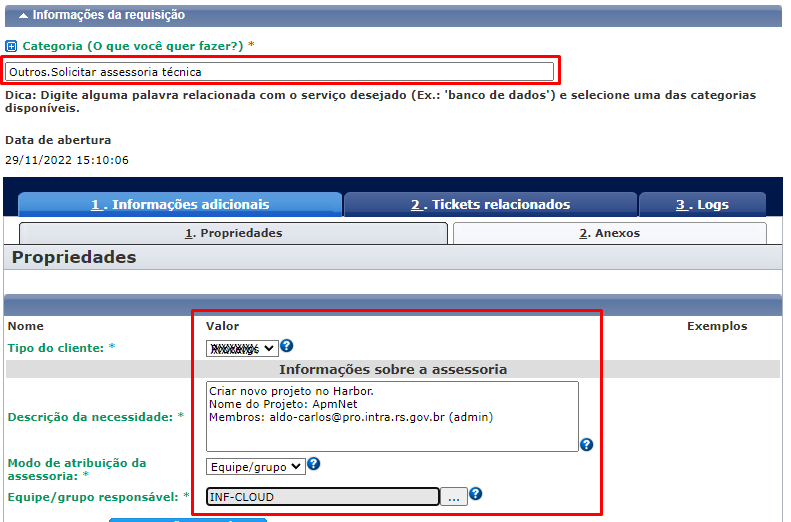
Para organizar o catálogo de imagens, o Harbor segmenta o repositório em uma estrutura de projetos, como mostrado na imagem abaixo:



O nome do projeto deve ser a sigla do sistema ou a sigla de identificação do cliente.

Na parte de permissionamento, cada projeto possui uma lista de usuários e administradores que terão acesso a inclui/excluir imagens do projeto.

Para solicitar a criação do Projeto ou acesso ao Harbor, acesse o **USD - Unicenter Service Desk**, na aba *Serviços técnicos*, pesquise pela opção **Outros.Solicitar assessoria técnica** e preencha os dados como a figura abaixo:



Utilizar o seguinte *template* para preencher o campo de descrição:

Criar novo projeto no Harbor.

Nome do Projeto: <sigla do sistema>

Membros: <Usuários com autorização de acesso>

Obs.: o usuário informado deve ser do domínio de produção.

## 3.3 – Pipeline

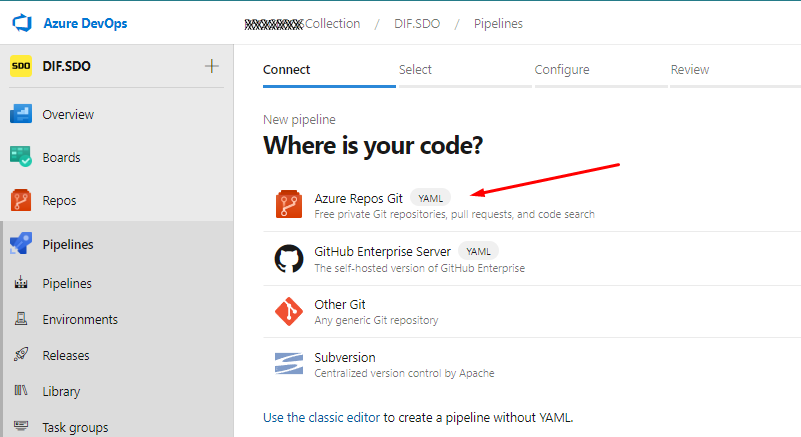
Este passo consiste no trabalho de criar a automatização do processo de geração da imagem da aplicação.

Para entender melhor esse processo, podemos dividi-lo nos seguintes passos abaixo:

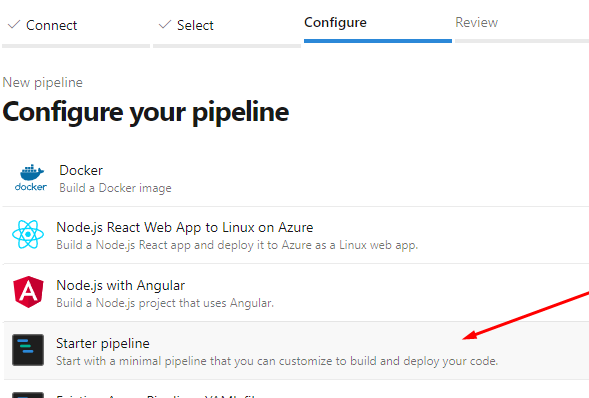
1. Baixar os códigos-fontes da aplicação (*git clone*)
2. Configurar parâmetros ou variáveis, como número de versão, Tag’s entre outros necessários para produzir a compilação de produção da aplicação.
3. Build da imagem (*docker build*). Aqui irá ser feita a compilação da aplicação propriamente dita assim como a geração da imagem Docker.
4. Publicar a imagem gerada no Harbor (*docker push*)

Dependendo da tecnologia da aplicação podemos ter pequenas variações ou passos adicionais, mas em síntese, os 4 passos descritos acima resumem um Pipeline de Build de uma aplicação para a Cloud.  
Este pipeline pode ser criado no Azure DevOps ou no Jenkins(para as aplicações Java, por exemplo).   
Assim como os Dockerfiles, também foram criados exemplos (ou *Templates*) para auxiliar no desenvolvimento dos Pipelines.  
Esses arquivos estão organizados e separados por tecnologia no seguinte repositório: <https://github.com/AldoCarlos/pa.templates>

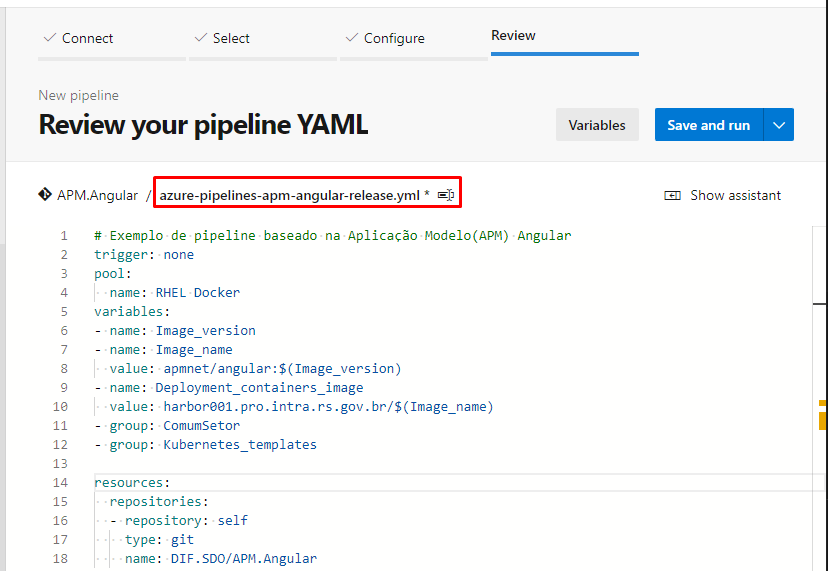
Para iniciar um novo pipeline no Azure DevOps, clicar no menu **Pipelines** e em seguida no botão **New Pipeline**. Escolher a opção **Azure Repos Git** como mostra a figura abaixo:



No passo seguinte selecione o repositório da aplicação e no próximo passo escolha a opção **Starter pipeline** como mostra a figura abaixo:



No editor YAML, clique no nome do arquivo YAML para renomear o arquivo e copie e cole o conteúdo do arquivo YAML de exemplo, como mostra a figura abaixo:



Revisar o conteúdo do arquivo, ajustando os parâmetros e variáveis e em seguida clicar no botão **Save and run**.

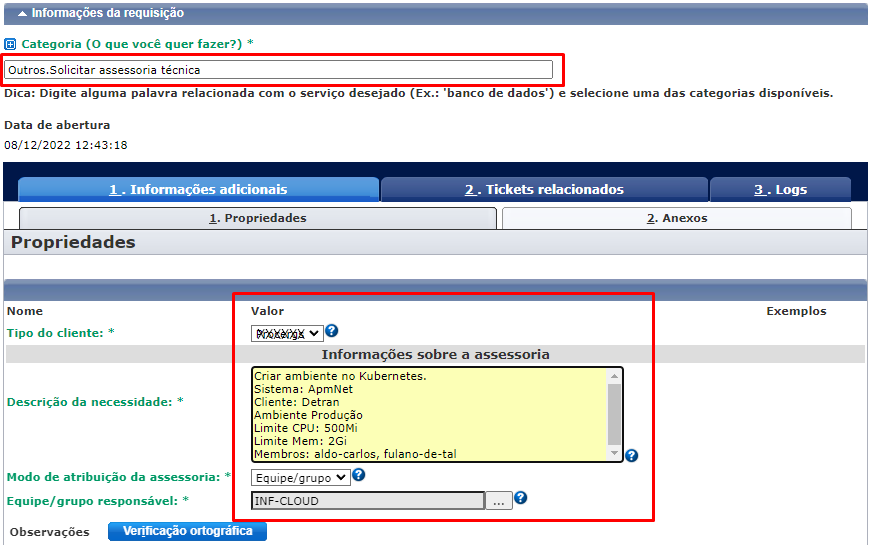
## 3.4 – Kubernetes

### 3.4.1 – Solicitar acesso

Para publicar uma aplicação no Kubernetes da Cloud interna da empresa, é necessário inicialmente solicitar a criação de um “espaço” em algum Cluster disponível. Esse espaço geralmente consiste na configuração de um ou mais **namespaces** dentro do Kubernetes. A criação de mais de um **namespace** pode ser utilizada para distinguir diferentes ambientes, por exemplo, para o sistema ApmNet podemos ter os seguintes nomes de **namespaces**:

* apmnet-des: para ambiente de desenvolvimento
* apmnet-hml: para o ambiente de homologação
* apmnet: para o ambiente de produção

Para fazer a solicitação descrita acima, é necessário acessar a ferramenta **USD - Unicenter Service Desk**, na aba *Serviços técnicos*, pesquise pela opção **Outros.Solicitar assessoria técnica** e preencha os dados como na imagem abaixo:



Utilizar o seguinte *template* para preencher o campo de descrição:

Criar ambiente no Kubernetes.

Sistema: <sigla do sistema>

Cliente: <sigla do cliente>

Ambiente <Produção|Desenvolvimento|Homologação>

Limite CPU: XXXMi

Limite Mem: XGi

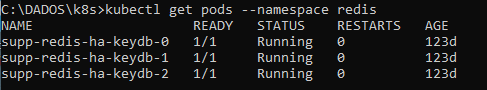
Membros: <Usuários com autorização de acesso>

### 3.4.2 – Configurar o 1º acesso

Após a execução e conclusão da requisição, será disponibilizado um arquivo *kubeconfig* com autorização de acesso no *Cluster/Namespace* criado.

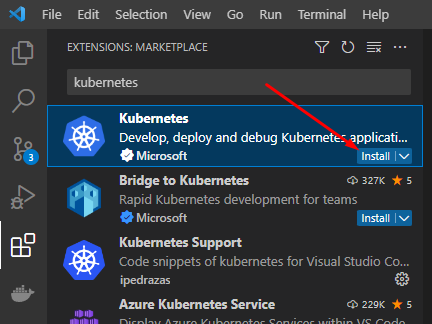
A maneira mais básica de acessar o *Kubernetes* é através da ferramenta de CLI **kubectl**. [Acessar este link](https://kubernetes.io/docs/tasks/tools/#kubectl) para fazer o download e instalação desta ferramenta.

Para utilizar o *kubectl* é necessário associar o arquivo *kubeconfig* recebido. Isso pode ser feito através de uma variável de ambiente chamada **KUBECONFIG**. Exemplo:  
  
> SET KUBECONFIG=C:\Path\to\Kubeconfig  
  
Feita essa configuração, o *kubectl* já pode ser executado. Segue abaixo um exemplo de um comando para listar os *Pods* de um *Namespace*:

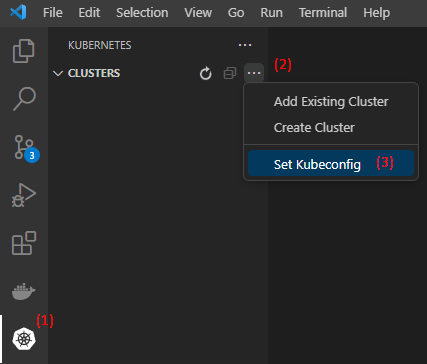


Usar o comando *kubectl –help* para obter ajuda sobre os comandos disponíveis ou então acessar o [guia de referência do site kubernetes.io](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands).

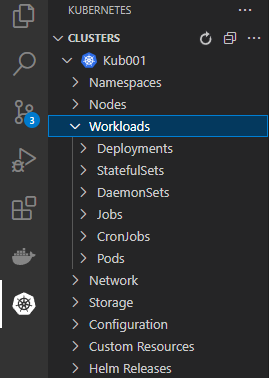
Além do *kubectl* existem diversas ferramentas disponíveis na internet para serem usadas como cliente de acesso ao *Kubernetes*. Uma bastante interessante, principalmente para a área de desenvolvimento, é uma extensão do *Visual Studio Code* para o Kubernetes. Para instalá-la clicar no menu “Extensions”, pesquisar por “Kubernetes” e em seguida clicar em Install, como mostra a figura abaixo:



Após a instalação um novo ícone do *Kubernetes* irá aparecer no menu lateral, clicar nele e em seguida no menu da figura abaixo, selecione a opção *Set kubeconfig* para informar o local do arquivo de configuração:



Em seguida já será possível visualizar a estrutura do *Kubernetes* para o *Cluster* disponível, como mostra a figura abaixo:



### 3.4.3 – Criar os objetos no kubernetes

Por fim, após ter todo o acesso necessário, já é possível criar as devidas configurações no *Kubernetes* para rodar a aplicação, como por exemplo, criar um *Deployment*, *Services*, *Secrets*, etc. Para facilitar a criação desses objetos, foi disponibilizado no repositório a seguir, arquivos de *Templates* para cada um deles:

<https://github.com/AldoCarlos/pa.templates/tree/master/kubernetes>

Nos arquivos do repositório acima, o *Template* de *Deployment* foi dividido em 2 arquivos já prevendo algumas diferenças de acordo com o tipo de aplicação:

* **DeploymentWeb.yaml**: para aplicações que expõe uma porta HTTP, por exemplo: API .Net ou Java, uma aplicação Angular.
* **DeploymentSvc.yaml**: para aplicações que não expõe uma porta HTTP, como por exemplo: rotinas *Batch*, aplicações do tipo *Workers*.

Um outro detalhe importante é que foi inserido no conteúdo dos arquivos de *Template* alguns parâmetros com um valor “curinga” representado pelos caracteres #{abc}#. A ideia é que esses parâmetros possam facilmente serem supridos por valores de variáveis durante o *Pipeline* do Azure Devops. Isso pode ser feito com a *Task* de “*Replace Tokens*”, por exemplo, durante o *Pipeline* de *Deploy* para o *Kubernetes*. Para mais detalhes, ver o capítulo [3 – Pipeline](#_3_–_Pipeline) deste guia.

Opcionalmente, a criação desses objetos no *Kubernetes* também pode ser feita inicialmente de forma manual. Para isso, pode ser utilizado o comando *kubectl* passando o arquivo YAML a ser criado, como por exemplo:

> kubectl apply -f C:\path\to\DeploymentWeb.yaml

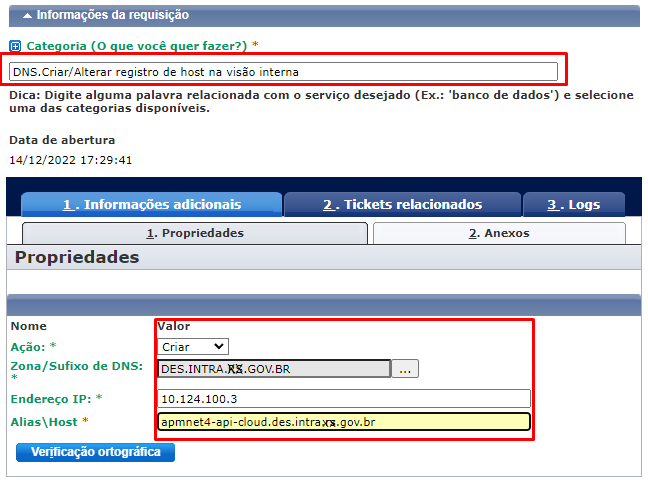
## 3.4 – DNS

Após fazer a publicação da aplicação no *Kubernetes*, é necessário “expor” a aplicação para acesso externo. Recomenda-se que isso seja feito através de um elemento ***Ingress*** do *Kubernetes*.

O *Ingress* nos permite configurar os protocolos HTTP/HTTPS, SSL/TLS, roteamento de tráfego, balanceamento de carga, controle de acesso, websocket e etc.

Ele também nos entrega uma porta e um número IP associado a uma URL. Este IP e URL devem ser registrados no serviço de rede (DNS) da empresa.

Para fazer a solicitação descrita acima, é necessário acessar a ferramenta **USD - Unicenter Service Desk**, na aba *Serviços técnicos*, pesquise pela opção **DNS.Criar/Alterar registro de host na visão interna** e preencha os dados como na imagem abaixo:



Para obter o endereço IP, pode-se utilizar o comando **kubectl** para listar as informações do **Ingress**. O comando é: kubectl get ingress -n <namespace>. A coluna “ADDRESS” irá conter o endereço IP que deve ser fornecido, como mostra a figura abaixo:

