# Guia para publicação de aplicações na Cloud Privada

**Introdução**

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

**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.

## 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 .

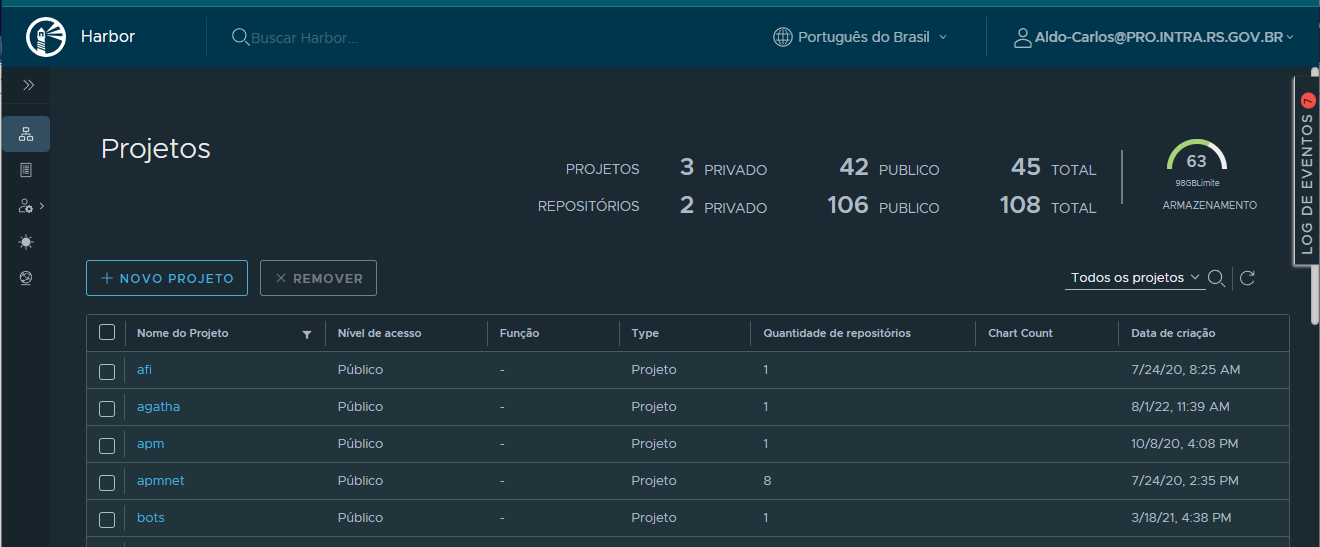
> docker run -d -p 8080:80 apm-angular:latest

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

## 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/>

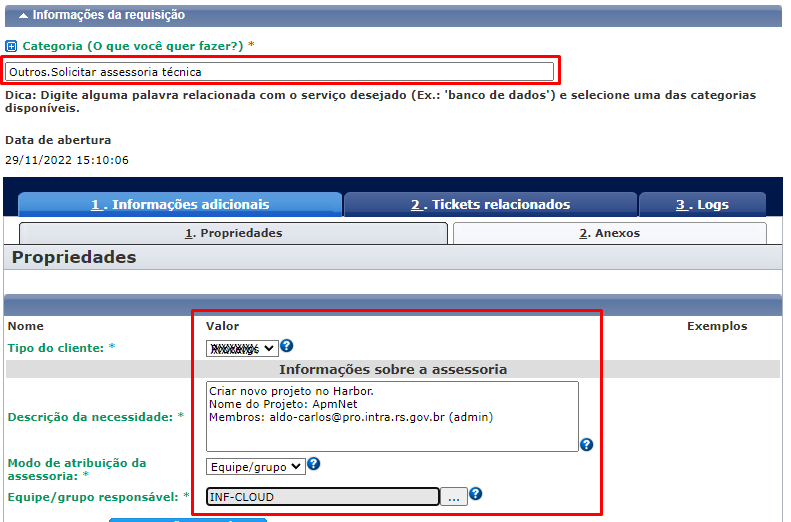
Para organizar o catálogo de imagens, o Harbor segmenta o repositório em uma estrutura de projetos, como mostrado na imagem abaixo:



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, acesse o USD, na aba *Serviços técnicos*, utilize a REQ **Outros.Solicitar assessoria técnica** e preencha os dados como abaixo:



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

## 3 – Pipeline

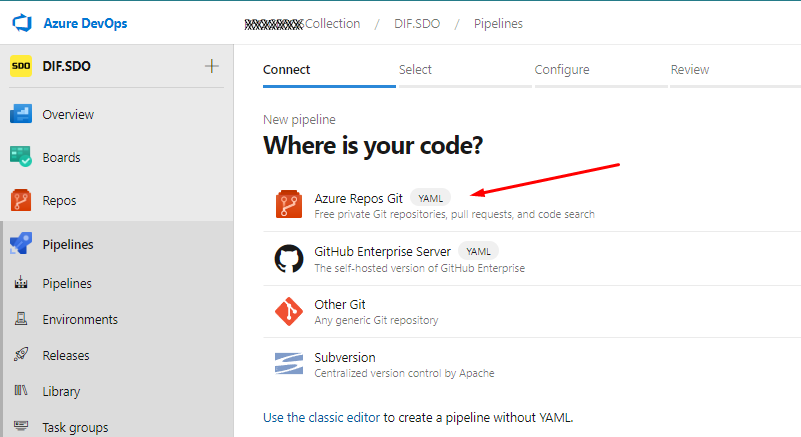
Este passo consiste no trabalho de criar a automatização do processo de geração da imagem da aplicação.

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

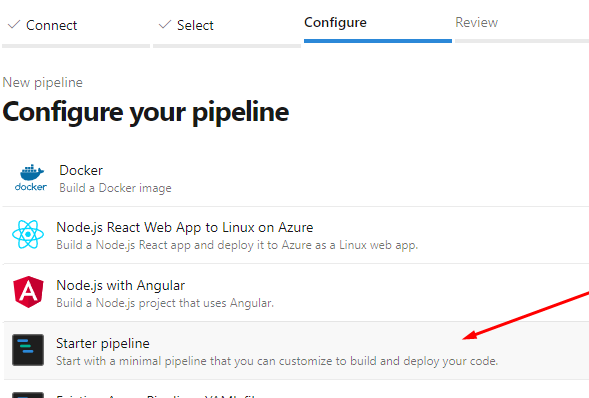
1. Baixar os códigos-fontes da aplicação (*git clone*)
2. Configurar parâmetros ou variáveis, como número de versão, Tag’s entre outros necessários para produzir a compilação de produção da aplicação.
3. Build da imagem (*docker build*). Aqui irá ser feita a compilação da aplicação propriamente dita 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>

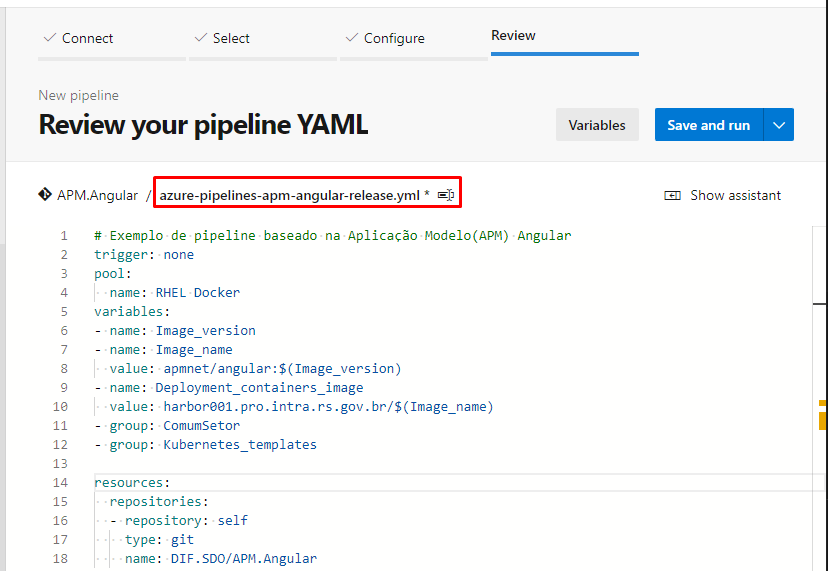
Para iniciar um novo pipeline no Azure DevOps, clicar no menu **Pipelines** e em seguida no botão **New Pipeline**. Escolher a opção **Azure Repos Git** como mostra a figura abaixo:



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**.

## 4 – Kubernetes

*<Desenvolver no Sprint 3>*

## 5 – DNS

*<Desenvolver no Sprint 3>*