**Documentação – Desafio de gerenciamento de Configuração**

**Data de Criação:** 11/11/2021

**Criado por:** Rafael Prado

**SUMÁRIO**

Esse procedimento tem como intuito orientar o provisionamento de uma Infraestrutura Cloud na Azure, e implantar uma aplicação web de linguagem GO em Docker Container.

**PRÉ REQUISITOS**

**Ferramentas:**  
Conta na Azure  
Terraform instalado em seu computador  
Visual Studio Code instalado em seu computador  
Git instalado em seu computador

**Repositório:**  
<https://github.com/rafael-prado10/myproject/tree/master>

**PROCEDIMENTO**

Passo 01: Baixe e descompacte em seu computador os arquivos do repositório:  
<https://github.com/rafael-prado10/myproject/tree/master>  
  
Passo 02: Abra o VS CODE, certifique-se de que esteja autenticado em sua conta Azure, abra a pasta “INFRA”.  
Tela de celular

Descrição gerada automaticamente

Autorize o VS CODE a abrir seus arquivos:  
Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Passo 03: O arquivo “Main.tf” constrói a Infraestrutura em conjunto com o que foi definido no arquivo “Variables.tf”. Defina suas variáveis no arquivo “Variables.tf” e salve-o teclando “Ctrl + s”.  
O “Main.tf” irá construir uma VM Linux e todo os recurso necessários para o funcionamento da mesma, a imagem escolhida é um CentoS 8 já com Docker incluso.  
Texto

Descrição gerada automaticamente

Passo 04: Abra um novo Terminal para executar o “Terraform” e provisionar a infraestrutura.  
Tela de celular com publicação numa rede social

Descrição gerada automaticamente

Passo 05: Inicie o “Terraform” com o seguinte comando:  
terraform init  
Texto

Descrição gerada automaticamente

Passo 06: Execute o seguinte comando:  
terraform plan  
Se não for encontrado nenhum erro, receberá a tela seguinte:  
Texto

Descrição gerada automaticamente

Passo 07: Execute o seguinte comando para criar os Recursos na sua conta Azure (Certifique-se de que já esteja autenticado):  
terraform apply  
Texto

Descrição gerada automaticamente

Confirme a criação dos recursos digitando “yes”:Texto

Descrição gerada automaticamente

Após o término, você já pode ver o ambiente criado em seu portal Azure:  
Tela de computador com jogo

Descrição gerada automaticamente

Passo 08: Os próximos passos será a implantação da aplicação GO, acesse o seu servidor através de conexão SSH pelo próprio VS CODE, utilize o seguinte comando:  
ssh user@ippublico  
  
(User = usuário que foi definido para criação do servidor / ippublico = IP Público do servidor).  
Digite “yes” para aceitar a conexão, em seguida digite a senha que foi definida para acesso no servidor.  
Texto

Descrição gerada automaticamente

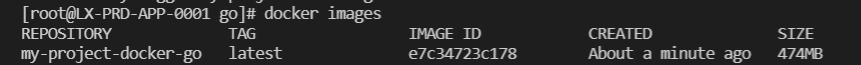
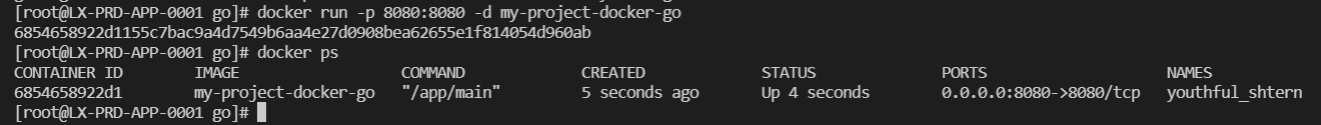
Passo 09: A pasta “App” do repositório contém os arquivos necessários para a criação da imagem Docker. O arquivo “main.go” é a aplicação, o arquivo “Dockerfile” contém as informações de configuração da imagem.   
Entre como root, utilizando o comando sudo su (será necessário digitar a senha novamente) após isso, crie uma pasta em qualquer diretório para que seja montado os arquivos para a criação da imagem que rodará no Docker Container, utilize o comando mkdir “nomedapasta”.

Transfira os arquivos para o seu servidor no diretório criado, ou somente copie o conteúdo do arquivo e crie diretamente no seu servidor:  
Texto

Descrição gerada automaticamente  
  
(Lembrando que o Docker já vem instalado e pronto para uso imagem do S.O que foi escolhido no provisionamento do servidor)

Passo 10: Com os arquivos dentro do mesmo diretório, você pode criar a sua imagem, utilize o seguinte comando:  
docker build -t my-project-docker-go   
Se os parâmetros estiverem corretos, será apresentado a seguinte tela:  
Texto

Descrição gerada automaticamente

Podemos ver a nossa imagem pronta para ser executada em container, digite o comando docker images:  
  
  
Passo 11: Com a imagem pronta para uso, iremos executar o seguinte comando:  
docker ruin -p 8080:8080 -d my-project-docker-go  
Em seguida, digite o comando docker ps, para ver o container em execução:  


Passo 12: Abra seu navegador, e acesse a aplicação pelo endereço:  
<http://ippublico:8080>  
Interface gráfica do usuário, Texto, Site

Descrição gerada automaticamente  
  
  
  
  
Passo 13: Agora que a sua aplicação está no ar, armazene o seu código em um repositório Git, onde possa fazer modificações e testes antes de atualizar o ambiente em produção, no exemplo será utilizado o “GitHub, crie uma conta através do link abaixo e crie também o seu primeiro repositório:  
https://github.com/  
  
Cerifique-se de que o GIT esteja instalado em seu computador local, navegue até a pasta onde estão os arquivos do seu projeto, e inicie o GIT clicando com o botão direito do mouse, em seguida, clique em “Git Bash Here”:

O GIT abrirá o Bash para linha de comando a partir da pasta do seu projeto:  
Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente  
  
Passo 14: Utilize os comandos “Git” abaixo:  
git init (inicia o git)

git config --global user.email “seu -email”, em seguida, git config –global user.name “seu nome de usuário” (identifica quem está comitando)

git status (visualiza os arquivos que precisam ser adicionados ao repositório)

git add . (adicione os arquivos para o seu controle de versão)  
  
git commit -m “Initial Commit” (realizar o commit do seu código para o repositório)  
  
git remote add origin + endereço do repositório (indica o repositório onde os arquivos vão ser adicionados)  
(exemplo: git remote add origin <https://github.com/rafael-prado10/project-example>)

git push --set-upstream origin master (caso não esteja autenticado será aberto uma nova janela para inserir a senha da conta)  
  
Pronto, seu código já está disponível no seu repositório GitHub, por padrão, ele sobe os arquivos na Branch “master”, que é a Branch do seu código atual em produção.  
Crie uma Branch com nome “staging”, utilize ela para os commits de testes, antes de enviar para a Branch master, assim, você tem um ambiente de testes separado e evita qualquer problema em produção.  
  
Passo 15: Provisione o Serviço “GitHub Actions” no servidor de aplicação, ele funciona como um CI/CD para automatizar as atualizações da aplicação diretamente no seu servidor, facilitando o processo de deploy do ambiente e controlando as versões diretamente pelo GIT.  
  
Passo 16: Para que o site responda em HTTPS, é necessário a instalação de certificado SSL no seu servidor, existem diversas maneiras e empresas certificadoras no mercado, onde você pode adquirir o seu certificado, registrado diretamente no seu domínio e DNS, você pode também utilizar um certificado Lets Encrypt, que é gratuito.   
  
Você pode instalar esse certificado no mesmo servidor em container diferente, ou até mesmo configurar direto no código da sua aplicação, veja as possibilidades existentes e a melhor maneira de utilizar em seu site.