Insper

# Computação em Nuvem

Cap. 6 - CI/CD - 4 Aulas

Raul Ikeda & Eduardo Marossi

**Grupo**:

Objetivos

1. Entender os conceitos básicos sobre Continuous Integration/Continuous Delivery (CI/CD).
2. Utilizar funções do Kubernetes como *deploy* de *pods* e gerenciamento de estado desejado.

Pré-requisitos:

1. Terminar o capítulo anterior (Container Orchestration)
2. Realizar a leitura sobre o Jenkins. [https://jenkins.io/].

Após a implantação do Kubernetes e uma tentativa de atualização de aplicação. Vamos montar um pipeline real onde a aplicação é montada e implantada automaticamente. Durante o processo ainda realizamos testes unitários e de integração. O ferramental utilizado nesse roteiro será: Kubernetes + Gitlab + Jenkins.

## Montagem do Ambiente

### Gitlab

Primeiramente vamos implantar e configurar o Gitlab, uma aplicação Git *Opensource*.

- Conecte na instância que gerencia o *Kubernetes*.
- Realizar um *deploy* da imagem do Gitlab Community Edition (gitlab/gitlab-ce:latest) na porta 80.
- Aguardar o serviço estar pronto:
  - $ watch kubectl get all
- Expor o serviço via *LoadBalancer* na porta 80. Reservar o endereço do *LoadBalancer*.
- Acessar o *GitLab*. Colocar uma senha padrão *cloud123*. O usuário padrão é *root*.
- Acessar a configuração do usuário e criar um *Token*:
  - Chamar de *jenkins*.
  - Colocar 1 ano de validade.
  - Marcar o escopo *API*.
  - Anotar o token gerado.
- Criar um repositório público chamado **aio-webserver**. Anotar o endereço git.

### Jenkins

O Jenkins é uma aplicação opensource para gerenciar CI/CD. Ele possui inúmeros *plugins* o que torna uma poderosa ferramenta. Existem diversas formas de integração entre repositórios e CI/CD. Vamos explorar uma delas.

- Vamos utilizar a imagem Docker oficial para o Jenkins: [https://hub.docker.com/r/jenkins/jenkins/]
- Ao invés de utilizar um comando como *kubectl run*, para realizarmos um *deployment*, pode ser utilizado um arquivo de configuração *YAML* descrevendo toda a estrutura do *deployment*, *pods* e serviços que desejamos criar no k8s.
- Criar um arquivo jenkins.yaml contendo o template do *deployment*:
  - Arquivo: [jenkins.yaml]
- Executar o comando:
- $ kubectl create -f jenkins.yaml

- Verifique se o *pod* do Jenkins inicia corretamente.
- Expor também o serviço e anotar o endereço do *LoadBalancer*.

1. Explique quais as vantagens e as desvantagens de utilizar um arquivo YAML para o *deployment* comparado ao utilizar o comando *kubectl create*.

**Configuração do Jenkins**

- Acessar o *dashboard* principal do Jenkins.

2. Como você acessou a senha do administrador?

- Seguir com a configuração fazendo a instalação recomendada dos Plugins.
- Coffee time!
- Continuar como administrador. Com isso o usuário será *admin* e a senha encontrada acima.
- Caso a página do Jenkins fique em branco após terminar a configuração ou realizar login. Altere o final do endereço no navegador para */restart* para forçar reniciar o servidor do Jenkins.
- Entrar no painel de configuração do Jenkins e instalar os *plugins*:
- GitLab
- GitLab Hooks
- Nas configurações do usuário, gerar um *API token* chamado *gitlab*. Reservar.
- Acessar as configurações gerais do sistema:
  - Configurar o GitLab utilizando o token criado no gitlab. Testar a conexão.
  - Desmarcar a opção: "Enable authentication for '/project' end-point".
  - Adicionar as credenciais do *Docker Hub* em *Registry credencials*. Chamar de *dockerhub*.
- Acessar as configurações de segurança global:
  - Marcar a opção: "Habilitar compatibilidade de proxy".
- Criar um novo *job* chamado *aio-webserver* do tipo Pipeline:
  - Marcar a opção: "Build when a change is pushed to GitLab".
  - Anotar o endereço *GitLab webhook URL*.
  - Adicionar o script de pipeline:
    - Arquivo: [pipeline_v1]
  - Salvar.

**Criando a aplicação**

De volta ao Gitlab, vamos começar a colocar código no repositório.

- No repositório, acessar a página *Integrations* em *Settings*:
  - Salvar um *Webhook* inserindo o *GitLab webhook URL* e o *token* criado no Jenkins.
  - Testar um *Push events*.
  - Se tudo certo, o Jenkins foi acionado automaticamente e executou a pipeline.
  - Verificar no log do dashboard do Jenkins a mensagem de erro.

3. Em qual estágio e por que deu esse erro?

- Clonar o repositório no seu computador.
- Adicionar o seguintes arquivos:
  - Arquivo: [senha.py]
  - Arquivo: [senha_teste.py]
- Testar localmente a aplicação usando:
  - $ python senha_teste.py
- *add*, *commit* & *push*.
- Verificar novamente o resultado do *Pipeline* no Jenkins.
- Fazer um breve teste:
  - Remover o comentário da variável rexes em senha.py.
  - Realizar o *commit* e *push*.
  - Verificar o *Pipeline*.
  - Reverter a alteração.

4. Você realizou uma *Pipeline* de *CI* com testes unitários. Contudo ainda faltam os testes de integração. Explique o que é teste de integração.

**Continuous Delivery**

Precisamos agora de um serviço web para expor a nossa aplicação via *endpoint*. Mas antes vamos modificar nosso pipeline para executar um arquivo Jenkinsfile que será colocado diretamente no repositório.

- Copiar o *Pipeline Script* atual. Reservar.
- Nas configurações da Pipeline, trocar para "Pipeline Script from SCM":
  - Configurar o endereço do repositório e as credenciais
- Adicionar um arquivo Jenkinsfile na raiz do repositório:
  - Copiar o *Pipeline Script*
  - Remover o *Stage Cloning*
- Verificar se o Jenkins conseguiu executar o *Pipeline*.

Agora vamos adicionar os outros elementos.

- Adicionar agora um servidor web assíncrono para servir o seu validador de senha:
  - Arquivo: [web.py]

Para implantar no *Kubernetes* precisamos de um imagem *Docker*. A ideia é fazer o próprio *Jenkins* montar a imagem e enviar para o Hub. Note que a biblioteca usada no webserver precisa ser instalada.

- Criar um Dockerfile que roda o servidor web:
- Testar o Dockerfile e a imagem docker localmente antes de realizar o *commit*.

Para realizar montagem da imagem e envio para o Hub:

- Criar um repositório no *Docker Hub* chamado *aio-webserver*.
- Trocar o *Pipeline Script*
  - Arquivo: [Jenkinsfile_v2]
- Rodar o *Pipeline*.

A mensagem de erro irá indicar que não existe a aplicação *Docker* instalada. Isso pelo simples fato do Jenkins tentar montar o *pipeline* localmente e sua imagem não possui realmente o *Docker*. Nesse ponto temos duas opções:

1. Criar uma nova imagem *Jenkins* com o *Docker* instalado. Opção meio complicada.
2. Rodar o pipeline em outro lugar. Ainda bem que temos o *Kubernetes* disponível.

- Instalar o *Plugin Kubernetes*.
- Nas Configurações Gerais, adicionar uma nova *Cloud Kubernetes*.
  - Atenção especial para a configuração: *Jenkins tunnel*.
- Modificar o Jenkinsfile para criar um pod Kubernetes:
  - Arquivo: [Jenkinsfile_v3]
- Incorporar no repositório o arquivo de criação do *pod* de montagem:
  - Arquivo: [build-pod.yaml]
- Rodar a *Pipeline* e verificar se a imagem foi enviada para o Hub.
- Fazer o *deploy* manualmente da aplicação criada no *Kubernetes*.
- Expor via *LoadBalancer* e testar.

Último passo: fazer com que a imagem do *pod* também seja atualizada na *Pipeline*.

- Alterar o arquivo de criação do *pod* para criar um container com *kubectl*
  - Arquivo: [build-pod_v2.yaml]
- Criar um *kubectl secret* com o arquivo de configuração
  - $ kubectl create secret generic kube-config --from-file=$HOME/.kube/config
- Adicionar a implantação o estágio final e voilà

```
stage('Deploying to Kubernetes') {
  steps {
    script {
      container('kubectl') {
        sh "kubectl set image deployment/aio-webserver hello-node=$registry:$BUILD_NUMBER"
      }
    }
  }
}
```

## Concluindo

1. Explique o conceito de *Continuous Integration/Continuous Delivery*.

## Mensagem final

Parabéns! Vocês finalizaram o curso de Computação em Nuvem com louvor. Atingiram um ponto onde poucos conseguiram chegar na disciplina. Depois de um semestre inteiro, ficam alguns pontos para reflexão ainda.
Vocês conseguem imaginar que o ferramental desse mundo é bem vasto. O que é visto na disciplina é apenas um pedacinho e em ambiente controlado.
Contudo nada impede de se especializarem ainda mais no assunto e nesse meio a experiência de execução é o melhor caminho para aprender. Também existem algumas certificações que podem ajudar nessa carreira.
*Keep walking*, estude, prototipe, teste e documente!