CENTRO UNIVERSITÁRIO IMEPAC

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LUCAS HENRIQUE DOS SANTOS AMARAL

JENKINS CD E CI

Araguari

2024

SUMÁRIO

1.INTRODUÇÃO

Nos últimos anos, no desenvolvimento de software, destacaram-se diversas plataformas e sistemas para automação. Neste contexto, um grande destaque foi o Jenkins, muito utilizado para integração contínua (CI) e Entrega contínua (CD), trazendo assim, agilidade, qualidade e eficiência no ciclo de vida do software.

Este trabalho tem como objetivo explorar o funcionamento, os benefícios e as aplicações práticas do Jenkins, demonstrando sua utilidade e praticidade para com o mundo do desenvolvimento.

No decorrer do trabalho, vamos falar sobre os conceitos básicos do Jenkins, mostrar exemplos de como usá-lo em tecnologias conhecidas, como Git, Docker e Kubernetes. E assim, apresentar uma visão clara de seu papel no desenvolvimento de software atual

2.O QUE É O JENKINS?

O Jenkins é um software open-source desenvolvido com o intuito de automação de sistemas. Ou seja, seu papel é realizar as tarefas que são repetitivas, permitindo ao desenvolvedor, focar-se em partes mais importantes na hora do desenvolvimento.

Ele é utilizado para **Integração Contínua (CI)**, ou seja, ele realiza verificações durante as alterações feitas para garantir que o sistema se mantenha em bom funcionamento. Outra de suas utilizações é em **entrega contínua (CD)**, que é o processo de colocar o código testado em produção de forma rápida e sem erros.

O Jenkins está no mercado desde 2004, inicialmente assumindo como um fork de uma outra ferramenta chamada Hudson, e foi lançado oficialmente em 2011.

3.PRINCIPAIS BENEFICIOS

O Jenkins oferece diversos benefícios para equipes de desenvolvimento. Automatizando tarefas repetitivas, compilando código, rodando testes e realizando deploys, o que reduz erros humanos e acelera o ciclo de desenvolvimento. Sua integração com diversas ferramentas, como Git, Docker e Kubernetes, permite uma maior flexibilidade e personalização dos fluxos de trabalho. Além disso, o Jenkins promove a Integração Contínua (CI) e a Entrega Contínua (CD), garantindo que o código seja testado e entregue de forma rápida e confiável. Isso resulta em um desenvolvimento mais eficiente, com menos falhas e maior qualidade no software.

4.ESTRUTURA BASICA

No Jenkins, os pipelines podem ser definidos de duas formas principais: **Declarativo** e **Scriptado**. Ambas as abordagens têm o mesmo objetivo — automatizar o processo de Integração Contínua (CI) e Entrega Contínua (CD) — mas diferem na forma de escrever e organizar o código.

4.1 PIPELINES DECLARATIVOS

O Pipeline Declarativo é uma forma mais simples e estruturada de definir um pipeline. Ele é focado em descrever o que você quer fazer, em vez de como fazer, e

normalmente é utilizado por iniciantes. Sua vantagem é uma estrutura clara e de fácil entendimento, porém isso a torna menos flexível para casos mais complexos. Conforme exemplo abaixo:

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                sh 'mvn clean install'
            }
        }
        stage('Test') {
            steps {
                sh 'mvn test'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploy realizado com sucesso!'
            }
        }
    }

}
```

4.2 PIPELINES SCRIPTADOS

O Pipeline Scriptado é mais flexível e permite que você escreva o código de forma mais detalhada. Ele permite maior controle sobre o processo, pois você pode adicionar lógica condicional, loops e outras scripts mais avançados. Conforme exemplo abaixo:

```
```

```
node {
    def branchName = 'main'
    def buildResult = ''

    try {
        stage('Preparação') {
            checkout scm
        }

        stage('Build') {
            buildResult = sh(script: 'mvn clean install', returnStatus: true)
            if (buildResult != 0) error "Falha na compilação!"
        }

        stage('Testes') {
            buildResult = sh(script: 'mvn test', returnStatus: true)
            if (buildResult != 0) error "Falha nos testes!"
        }

                                        .

                                        .

                                        .
```

5.JENKINS NA PRATICA

Em um fluxo típico, o Jenkins é configurado para monitorar alterações no código-fonte, como commits em repositórios Git. Quando uma nova alteração é detectada, o Jenkins automaticamente executa uma série de etapas, como compilar o código, rodar testes automatizados e, caso tudo esteja correto, realizar o deploy da aplicação. Essa automação não só acelera o ciclo de desenvolvimento, mas também garante maior qualidade e confiabilidade no software, reduzindo erros humanos e aumentando a eficiência das equipes. Além disso, o Jenkins pode ser facilmente integrado com outras ferramentas,

como Docker, Kubernetes e SonarQube, tornando-o uma solução flexível e poderosa para diversos cenários de desenvolvimento.

6.CONCLUSÃO

O Jenkins se consolidou como uma ferramenta fundamental para automação de processos no desenvolvimento de software, especialmente no contexto de Integração Contínua (CI) e Entrega Contínua (CD). Sua capacidade de automatizar tarefas repetitivas, como compilação, testes e deploy, não só aumenta a eficiência das equipes, mas também melhora a qualidade do código e reduz a ocorrência de erros. A flexibilidade do Jenkins, que permite integração com diversas ferramentas e a personalização de pipelines, torna-o uma solução robusta para diferentes necessidades de desenvolvimento. Portanto, o Jenkins é uma peça chave para times que buscam otimizar seus fluxos de trabalho e entregar software de alta qualidade de forma rápida e confiável.