Universitatea Tehnică a Moldovei

INTEGRAREA PROIECTELOR PE BAZA GITLAB ȘI KUBERNETES

Student:

Alexandru ȘOLOPA

Conducător:

conf.univ., dr. Valentin NEGURĂ

Chişinău - 2020

АННОТАЦИЯ

для магистерской работы на тему «Интеграция проектов на базе Gitlab и Kubernetes», разработанную студентом Шолопа Александр, группы CRI-191M

Целью данной работы является разработка практического мануала для использования непрерывной интеграции и непрерывной доставки в промышленных условиях.

Основным компонентом для данного примера был выбран программный продукт Gitlab, который оснащен набором нативных компонентов, позволяющих организовать в полной мере непрерывную интеграцию, построенную при использовании симбиоза декларативного и императивного подходов. Также непрерывная доставка настраивается с использованием сторонних интеграций, например подключения к Gitlab кластера Kubernetes, которые позволяют, используя веб интерфейс Gitlab, контролировать и настраивать инфраструктуру вокруг приложения.

Данная работа реализует поддержку непрерывной интеграции и непрерывной доставки для тестового приложения, написанного на языке Java. Внутри интеграционного скрипта используются специфичные для Java приложения инструкции сборки и тестирования приложения, а также упаковки в docker контейнер. Но также по аналогии, учитывая специфичность каждого отдельно взятого проекта, есть возможность построить систему для автоматизации процесса доставки функционала приложения от состояния исходного кода и до конечного пользователя.

Глава 1: описывает общие понятия о непрерывной интеграции и непрерывной доставки, а также аргументацию использования тех или иных технологий.

Глава 2: аккумулирует использованные в работе технологии и программные продукты, их преимущества и недостатки, а также уточняется какие компоненты данных продуктов были использованы в работе.

Глава 3: представляет собой имплементацию системы непрерывной интеграции и непрерывной доставки для тестового приложения.

Ключевые слова: непрерывная интеграция, непрерывная доставка, Gitlab, Kubernetes, Java, Spring Framework, Docker, Minicube.

ADNOTARE

la teza de masterat cu tema ” Integrarea proiectelor pe baza Gitlab și Kubernetes”
a st. gr. CRI-191, ȘOLOPA Alexandru

Scopul acestei lucrări este de a dezvolta un manual practic pentru utilizarea integrării continue și a livrării continue într-un cadru industrial.

Componenta principală pentru acest exemplu a fost produsul software Gitlab, care este echipat cu un set de componente native care vă permit să organizați complet integrarea continuă, construită utilizând o simbioză a abordărilor declarative și imperative. Livrarea continuă este, de asemenea, configurată utilizând integrări terțe, cum ar fi conectarea la Gitlab a clusterului Kubernetes, care permit, utilizând interfața web Gitlab, să controleze și să configureze infrastructura din jurul aplicației.

Această lucrare implementează integrarea continuă și suportul de livrare continuă pentru o aplicație de test scrisă în limbajul Java. În cadrul scriptului de integrare, instrucțiunile specifice Java sunt utilizate pentru a construi și testa aplicația, precum și pentru a o împacheta într-un container de andocare. Dar, de asemenea, prin analogie, luând în considerare specificitatea fiecărui proiect individual, este posibil să se construiască un sistem care să automatizeze livrarea funcționalității aplicației din starea codului sursă către utilizatorul final.

Capitolul 1: descrie conceptele generale de integrare continuă și livrare continuă, precum și raționamentul pentru utilizarea anumitor tehnologii.

Capitolul 2: acumulează tehnologii și produse software utilizate în muncă, avantajele și dezavantajele acestora și clarifică, de asemenea, ce componente ale acestor produse au fost utilizate în muncă.

Capitolul 3: introduce implementarea unui sistem de integrare continuă și livrare continuă pentru o aplicație de testare.

Cuvinte cheie: integrare continuă, livrare continuă, Gitlab, Kubernetes, Java, Spring Framework, Docker, Minicube.

ANNOTATION

to the master thesys on topic "Project integration based on Gitlab and Kubernetes" of st. gr. CRI-191M, ȘOLOPA Alexandru

The main goal of this work is to develop a practical manual for using continuous integration and continuous delivery in an industrial setting.

The main component for this example was the Gitlab software product, which is equipped with a set of native components that allow you to fully organize continuous integration, built using the symbiosis of declarative and imperative approaches. Continuous delivery is also configured using third-party integrations, such as connecting to Gitlab of the Kubernetes cluster, which allow, using the Gitlab web interface, to control and configure the infrastructure around the application.

This work implements continuous integration and continuous delivery support for a test application written in the Java language. Inside the integration script, Java-specific instructions are used to build and test the application, as well as package it into a docker container. But also by analogy, taking into account the specificity of each individual project, it is possible to build a system to automate the process of delivering application functionality from the state of the source code to the end user.

Chapter 1: describes the general concepts of continuous integration and continuous delivery, as well as the reasoning for the use of certain technologies.

Chapter 2: accumulates technologies and software products used in the work, their advantages and disadvantages, and also clarifies which components of these products were used in the work.

Chapter 3: introduces a continuous integration and continuous delivery system implementation for a test application.

Keywords: continuous integration, continuous delivery, Gitlab, Kubernetes, Java, Spring Framework, Docker, Minicube.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире все более актуальным становится вопрос баланса качества поставляемого на рынок программного обеспечения и скорости разработки продукта. Благодаря глобализации и распространению интернет технологий, появилась жесткая конкуренция в сфере разработки программного обеспечения, так как программное обеспечение давно вышло за рамки решения локальных проблем и готово предоставлять продукт или услуги всему миру, а также благодаря этому и разработка может вестись из разных уголков мира. Также само программное обеспечение претерпело изменения под стать растущим аппетитам рынка и сейчас это далеко не результат работы пары программистов, а десятков, сотен и даже тысяч. Для слаженной работы таких больших коллективов необходимо было разработать план взаимодействий внутри компаний, который бы учитывал вклад каждого отдельного члена команды и также максимизировал его продуктивность.

В таких условиях скорость разработки и ее качество становится острой проблемой компаний. И так как спрос рождает предложение – на рынке появились системы учета ошибок и управления проектами. Такое программное обеспечение помогло организовать рабочий процесс больших групп разработчиков, а также ускорить процесс разработки. Вместе с этим получил своё распространение Agile подход к разработке, который подразумевает под собой предоставление части функционала конечному пользователю и параллельно заниматься разработкой оставшегося функционала. Но так как конечной целью разработки является не только написание кода, а еще предоставление программного продукта конечным пользователям, что в свою очередь также является трудоемким и время затратным процессом, особенно если учитывать стабильный рост сложности высокотехнологических систем. Всё это приводит к тому, что разработка программного обеспечения происходит относительно быстро, но вот доставляется новый функционал до конечного пользователя с большим опозданием. Так как решение данной проблемы не являлось тривиальной задачей, данная задача породила множество вариаций ее решений, которые в последствии объединили под общим названием Непрерывной интеграции и Непрерывной доставки - CI/CD (Continuous Integration/ Continuous Delivery), обозначающих набор практик и принципов, которыми руководствовались те или иные разработчики для ускорения их процесса разработки.

И так как Time-to-market(TTM) является главным показателем, нужны инструменты, которые позволят следить за состоянием проекта, автоматически тестировать его после любых изменений, упаковывать готовое приложение и разворачивать на прод сервере, то есть автоматизировать всё, что можно автоматизировать в цепочке от изначального коммита кода и до запуска функционала на продакшн сервере.

БИБЛИОГРАФИЯ

1. https://docs.gitlab.com/
2. https://kubernetes.io/ru/docs/home/
3. https://github.com/kubernetes/minikube/releases
4. https://about.gitlab.com
5. Марко Лукша. *Kubernetes в действии.* 2019.
6. Арундел Джон, Домингус Джастин. *Kubernetes для DevOps: развертывание, запуск и масштабирование в облаке.* 2020.
7. O'Reilly. *GitLab Cookbook.* 2015.
8. Ибрам Билджин, Хасс Роланд. *Паттерны Kubernetes: Шаблоны разработки собственных облачных приложений*. 2020.
9. Сайфан Джиджи. *Осваиваем Kubernetes. Оркестрация контейнерных архитектур*. "Издательский дом ""Питер""", 26 дек. 2018 г. 400с.