РАЗРАБОТКА КОНФИГУРАЦИИ CI/CD ДЛЯ АВТОМАТИЗАЦИИ РАЗВЁРТЫВАНИЯ И УПРАВЛЕНИЯ ПРИЛОЖЕНИЯМИ

DEVELOPING A CI/CD CONFIGURATION FOR AUTOMATING DEPLOYMENT AND APPLICATION MANAGEMENT

УДК 004

Литвяков Дмитрий Сергеевич, студент магистратуры, МИРЭА – Российский технологический университет, г. Москва

Litvyakov Dmitry Sergeevich, Graduate Student, MIREA - Russian Technological University, Moscow, e-mail: litds@mail.ru

Аннотация

Многие учреждения здравоохранения в ходе своей деятельности пользуются специфичным и узкоспециализированным программным обеспечением. В связи с этим часто в организациях присутствуют разработчики, отвечающие за поддержание работы приложений, их модернизацию и совершенствование. Для проведения каждого испытания работы приложения разработчик должен вручную проводить рутинные действия по сборке, тестированию и развёртыванию. Всё это занимает много времени, требует внимательности человека и определённой квалификации. Решением данной ситуации является автоматизация описанного выше процесса. В статье приведены теоретические сведения об автоматизации развёртывания и управления приложениями, сформированы предложения о необходимости применения различных программных средств для достижения поставленной задачи, произведена разработка конфигурации CI/CD для автоматизации развёртывания и управления приложениями, описано создание виртуальной

среды для тестового стенда и продемонстрирована работа описанной конфигурации.

Annotation

Many health care institutions use specific and highly specialized software in the course of their activities. In this regard, there are often developers in organizations who are responsible for maintaining the operation of applications, their modernization and improvement. For each test of the application, the developer must manually perform routine assembly, testing, and deployment activities. All this takes a lot of time, requires the attention of a person and a certain qualification. The solution to this situation is to automate the process described above. The article provides theoretical information about the automation of deployment and application management, makes suggestions about the need to use various software tools to achieve the task, develops the CI/CD configuration for automating deployment and application management, describes the creation of a virtual environment for the test bench, and demonstrates the operation of the described configuration.

Ключевые слова: непрерывная интеграция, непрерывное развёртывание, CI/CD, здравоохранение, docker, git.

Keywords: continuous integration, continuous deployment, CI/CD, healthcare, docker, git.

В данной работе рассматривается одна из систем ФГБУ «НМИЦ ДГОИ им. Дмитрия Рогачева» Минздрава России. После подробного ознакомления с системой был выявлен важный недостаток, который предлагается решить с помощью дальнейших рекомендаций. Эта система была разработана специально под нужды исследуемого учреждения, она постоянно дорабатывается, обновляется и модернизируется. Однако процесс внедрения новых разработок замедляется за счёт ручной сборки, тестирования и развёртывания программного кода.

Для решения проблемы рекомендуется внедрить в системе поддержку конфигурации «Continuous integration / Continuous delivery» (CI/CD), что в

переводе означает — «Непрерывная интеграция / Непрерывное развёртывание», что позволит автоматизировать развёртывание и управление приложениями.

Непрерывная интеграция (CI) применяется для проведения интеграции и выявления возможных ошибок. Одним из наиболее выделяющихся элементов CI является этап автоматизированной сборки приложения [1]. Для выполнения непрерывной интеграции разработчики приложений делают частую выгрузку своих работ. Для возможности скорого обнаружения ошибок каждая интеграция должна осуществляться автоматизировано. Благодаря реализации CI можно значительно снизить трудоёмкость работы разработчика и всего проекта в целом [2]. Непрерывное развёртывание (CD) является автоматизированной отправкой кода из репозитория на удалённый сервер

Для реализации непрерывной интеграции потребуются следующие компоненты: рабочий компьютер разработчика, система контроля версий, система управления репозиториями кода, платформа управления контейнерами.

На рабочем компьютере разработчика хранится исходный кода проекта, для которого реализуется конфигурация CI/CD. На компьютере должна быть установлена система контроля версий для возможности внесения изменений в проект и последующем их отслеживании. Наиболее распространённой системой контроля версий является Git.

Далее необходима система управления репозиториями кода — это место куда будут отправляться все изменения проекта, внесённые разработчиком. Самыми популярными являются GitHub, Directual, Beanstalk, Bitbucket и GitLab [3]. Последний на момент написания работы используется в исследуемом учреждении, поэтому в дальнейшей работе будет разрабатываться конфигурация CI/CD применительно к GitLab.

Чтобы исключить возможность влияния каких-либо внешних факторов на запуск сборки и тестирования приложения потребуется изолированное окружение, в котором будут запускаться эти процессы. Для этого применяется платформа управления контейнерами. Контейнеры являются неизменяемыми. После определения контейнера его можно повторно создать и запустить точно

таким же образом [4]. Существует несколько платформ управления контейнерами, так в их число входят: Docker, Kubernetes, CoreOS, Etcd, rkt, Mesos. Для данной работы была выбрана платформа Docker, так как она является наиболее распространённой, имеет обширную документацию, удобна в использовании и имеет интеграцию с GitLab.

Для работы с Docker написан файл «Dockerfile», который представляет собой инструкцию, в соответствии с которой будет производиться сборка образа приложения для его последующего запуска (Листинг 1) [5].

Листинг 1 — Файл «Dockerfile»

```
FROM ruby:2.3.1
RUN mkdir -p /usr/src/lib
WORKDIR /usr/src/lib
COPY Gemfile Gemfile.lock /usr/src/lib/
RUN gem install bundler -v 1.15.1
RUN bundle install
RUN curl -sL https://deb.nodesource.com/setup_14.x | bash -
RUN apt-get update && apt-get install -y --force-yes\
    nodejs \
    mysql-client \
    python-pip \
    libmysqlclient-dev
COPY package.json /usr/src/lib/
RUN npm install && npm audit fix
COPY . .
EXPOSE 3000

CMD ["bundle", "exec", "rails", "server", "-b", "0.0.0.0"]
```

В «Dockerfile» происходит скачивание образа языка Ruby, создаётся рабочее пространство, копируются необходимые файлы, в соответствии с которыми далее происходит установка требуемых компонентов и пакетов, а также устанавливаются зависимости.

Далее был написан файл для запуска конвейера «.gitlab-ci.yml». Этот файл имеет синтаксис на языке YAML и содержит сценарии, которые объединяются в задачи. В данном файле описаны задачи для сборки, тестирования и развёртывания приложения.

Листинг 2 — Файл «.gitlab-ci.yml»

```
stages:
  - build
  - test
  - deploy
```

Продолжение Листинга 2

```
Build:
  image: docker:latest
```

```
  stage: build
  services:
    - docker:dind
  script:
    - docker login -u "$CI_REGISTRY_USER" -p "$CI_REGISTRY_PASSWORD"
$CI_REGISTRY
    - docker build --pull -t "$CI_REGISTRY_IMAGE${tag}" .
  rules:
    - if: $CI_COMMIT_BRANCH
      exists:
        - Dockerfile
  tags:
    - runner

Test:
  stage: test
  needs:
    - job: Build
  script:
    - echo "Testing"
  tags:
    - runner

Deploy to server:
  stage: deploy
  needs:
    - job: Test
  script:
    - git remote -v
    - git branch
    - git checkout master
    - cd /home/user/Projects/ProjectName
    - git pull origin master
  tags:
    - server
```

В сборке команда «docker build» запускает сборку проекта по файлу «Dockerfile». Происходит это на компьютере разработчика благодаря тегу runner. Далее идёт тестирование, в котором разработчик может поместить необходимые программы для тестирования приложения. После прохождения этих этапов начинается развёртывание на сервере приложений по тегу server. Командой git pull происходит получение всех изменений из удалённого репозитория GitLab.

Для интеграции CI/CD в исследуемом учреждении потребуется произвести следующие действия:

- на компьютере разработчика:
  - установить систему контроля версий Git;
  - установить Docker;
  - установить GitLab Runner с тегом runner;
- на удалённом сервере:

- установить систему контроля версий Git;
- установить GitLab Runner с тегом server.

GitLab Runner — это специальный агент, который устанавливается на устройстве и запускает конвейер используя файл «.gitlab-ci.yml».

Для демонстрации работы были созданы две виртуальные машины: одна для имитации компьютера разработчика, другая — сервер.

На компьютере разработчика был установлен Git, Docker, а также установлен и зарегистрирован GitLab Runner.

При запросе системы указаны следующие дополнительные данные:

– tags: runner (тег для идентификации агента в «.gitlab-ci.yml»);
– executor: docker (агент для запуска команд использует Docker).

На стороне сервера были проведены аналогичные действия по настройке GitLab Runner:

– tags: server;
– executor: shell (агент для запуска команд использует оболочку системы).

В результате выполнения всех описанных выше действий получается настроенный и рабочий конвейер CI/CD.

Для того, что убедиться в работе конвейера с компьютера разработчика следует внести какие-либо изменения в файлы проекта и выполнить следующие команды в командной строке (Листинг 3):

Листинг 3 — Команды для запуска конвейера

```
git add -A
git commit -m "Fix bugs"
git push origin master
```

В листинге описаны команды для добавления изменённых файлов, добавления комментария и отправки в удалённый репозиторий GitLab.

В результате как только файлы попадают в GitLab, автоматически начинает работу конвейер и выполняет задачи, описанные в файле «.gitlab-ci.yml». Происходит сборка, тестирование и развёртывание приложения на удалённом сервере.

Литература

1. Арланов И. В. Основные методы и компоненты систем непрерывной интеграции в ИТ-проектах // Новое слово в науке и практике: гипотезы и апробация результатов исследований. 2013. №8.
2. Хамбл Джез, Фарли Дейвид. Непрерывное развертывание ПО: автоматизация процессов сборки, тестирования и внедрения новых версий программ.: Пер. с англ. — М.: ООО «И.Д. Вильямс», 2011. — 432 с.: ил. — Парал. тит. англ.
3. Сайт аналитического агентства, исследующего рынок информационных технологий, интерактивного маркетинга и сопутствующих сервисов «Tagline» / Рейтинг сервисов-репозиториев для хранения кода 2018. — Режим доступа: https://tagline.ru/source-code-repository-rating. Загл. с экрана. — Яз. рус.
4. Электронное руководство по архитектуре приложений .NET компании «Microsoft» / Использование контейнеров и оркестраторов. — Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/architecture/cloud-native/leverage-containers-orchestrators. Загл. с экрана. — Яз. рус.
5. Электронная документация по сборке в системе «Docker» / Dockerfile reference. — Режим доступа: https://docs.docker.com/engine/reference/builder/. Загл. с экрана. — Яз. англ.

Literature

1. Arlanov I. V. Basic methods and components of continuous integration systems in IT projects / / A new word in science and practice: hypotheses and testing of research results. 2013. No. 8.
2. Humble Jez, Farley David. Continuous software deployment: automation of the processes of assembly, testing and implementation of new versions of programs.: Trans. from English-M.: LLC "I. D. Williams", 2011. - 432 p.: ill. - Paral. tit. en.
3. Website of the analytical agency that researches the market of information technologies, interactive marketing and related services "Tagline" / Rating of

services-repositories for storing code 2018. - Access mode: https://tagline.ru/source-code-repository-rating. Title from the screen. - Lang. ru.

4. Electronic guide to Application architecture Microsoft .NET / Using containers and orchestrators. - Access mode: https://docs.microsoft.com/ru-ru/dotnet/architecture/cloud-native/leverage-containers-orchestrators. Title from the screen. - Lang. ru.
5. Electronic documentation on the assembly in the "Docker" system / Dockerfile reference. - Access mode: https://docs.docker.com/engine/reference/builder/. Title from the screen. - Lang. en.