PLAN WDROŻENIA APLIKACJI WIREAPP DESKTOP

Z WYKORZYSTANIEM JENKINSA, DOCKERA ORAZ GIT'A

BARTOSZ BŁYSZCZ

INDEKS: 401928

2022-05-07

Spis treści

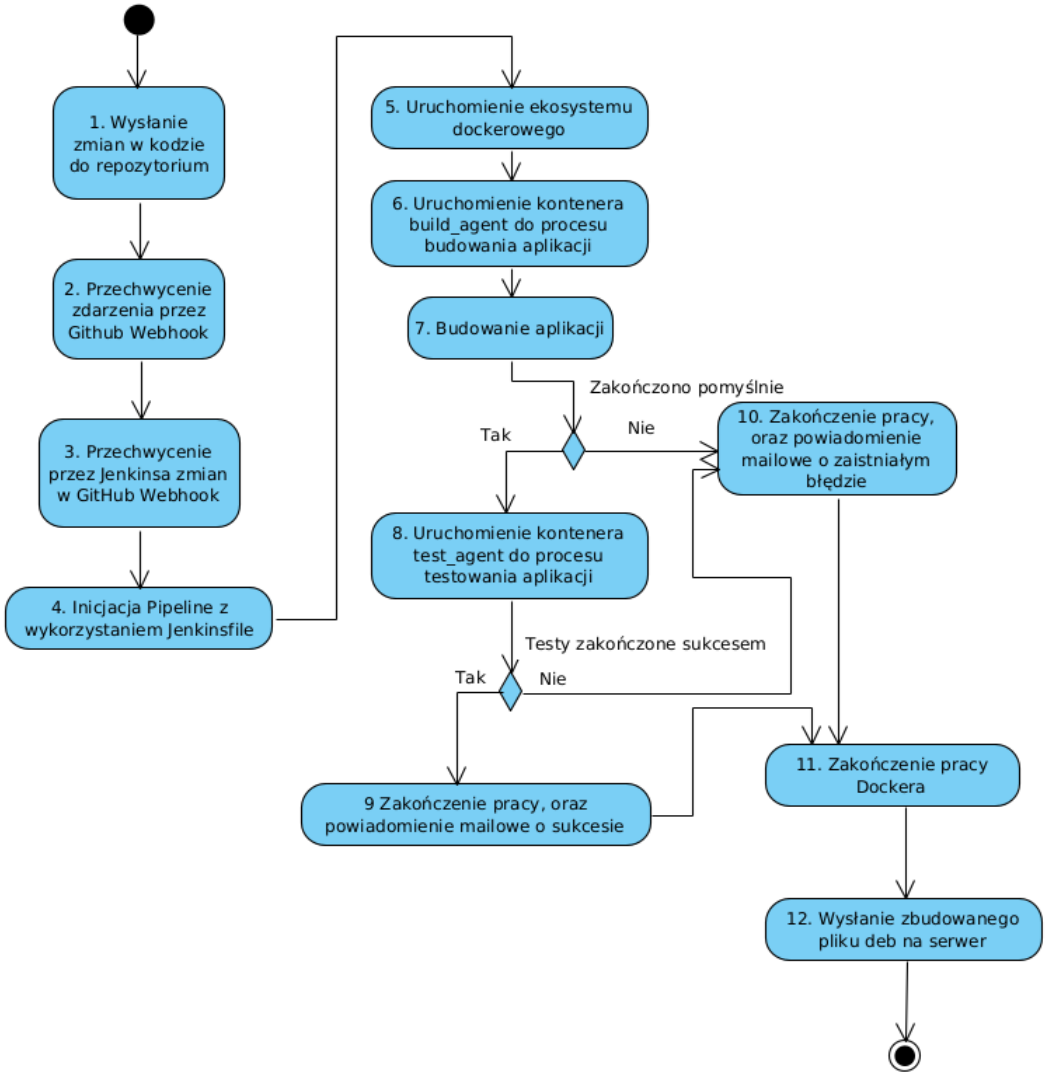
1. Plan wdrożenia

1.1. Wykorzystane technologie oraz narzędzia

Technologia	**Opis**
Docker	Narzędzie służące do wirtualizacji na poziomie systemu operacyjnego
Docker Registry	Narzędzie służące do przechowywania obrazów dockerowych *(użyto docker hub)*
Git	Rozproszony system kontroli wersji
Github	Serwer hostingowy, służący do przechowywania zdalnych repozytoriów
Jenkins	Serwer służący do automatyzacji związanej z tworzeniem oprogramowania

Tabela 1.1. Opis wykorzystanych technologii

1.2. Schemat wdrożenia aplikacji



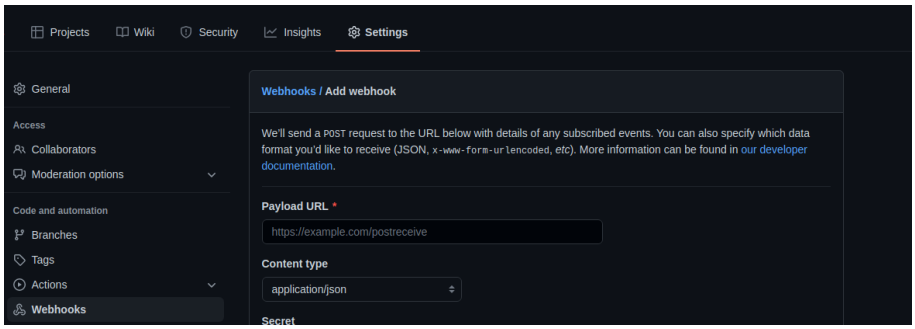
Rys. 1.1. Diagram aktywności, wdrożenia aplikacji

1.2.1. Wysłanie zmian w kodzie do repozytorium

Po zakończonej pracy nad kodem, programista za pomocą narzędzia **git** zapisuje swoje zmiany oraz przesyła je na repozytorium w serwisie GitHub (https://github.com/Suvres/wire-desktop).

1.2.2. Przechwycenie zdarzenia przez Github Webhook

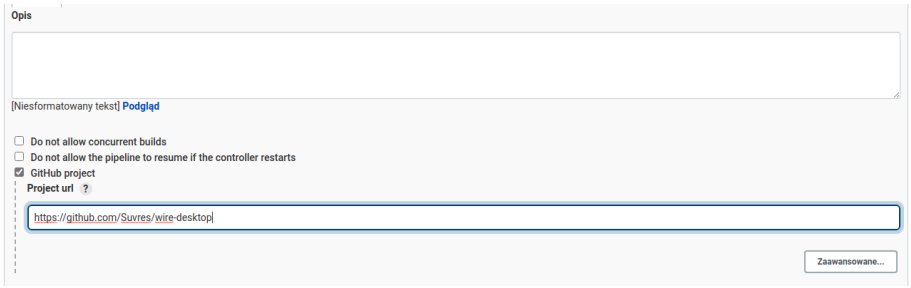
Do repozytorium powinien zostać dodany GitHub WebHook. Dzięki temu, po każdej wysłanej zmianie kodu do repozytorium wystąpi żądanie POST, wysłane na konkretny adres URL podany w konfiguracji. Adres ma schemat: **<URL Serwera Jenkins>/github-webhook/**. Przykład: **https://123.23.12.1:8080/github-webhook/**.



Rys. 1.2. Konfiguracja WebHook w serwisie Github

1.2.3. Przechwycenie przez Jenkinsa zmian w GitHub Webhook

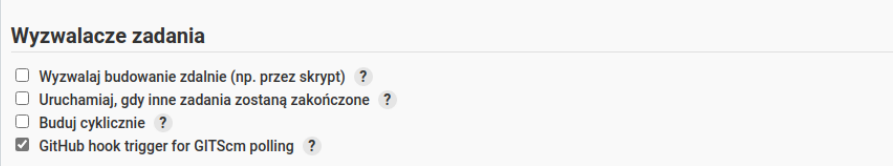
W Jenkinsie należy utworzyć nowy projekt **Pipeline** i w jego konfiguracji skonfigurować zakładkę "**Repozytorium kodu**", przez podanie odpowiedniego adresu URL do oczekiwanego repozytorium. W tym przypadku jest to adres aplikacji wireapp-dekstop: **https://github.com/Suvres/wire-desktop.git**.



Rys. 1.3. Zakładka ”Repozytorium kodu” w konfiguracji projektu w serwisie Jenkins

Następnym krokiem jest zaznaczenie w zakładce **Wyzwalacze zadania** *GitHub hook*

trigger for GITScm polling, dzięki czemu będzie mogła zaistnieć komunikacja między serwisami Jenkins i Github.

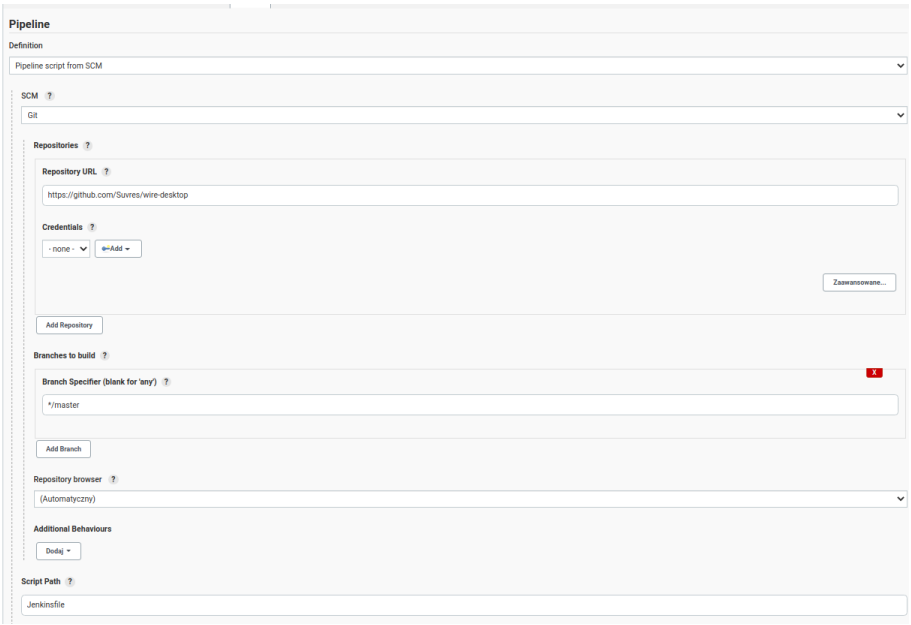


Rys. 1.4. Zakładka ”Wyzwalacze zadania” w konfiguracji projektu w serwisie Jenkins

Dzięki czemu, po otrzymaniu żądania POST z serwisu Github może nastąpić uruchomienie procesu Pipeline.

1.2.4. Inicjacja Pipeline z wykorzystaniem Jenkinsfile

Cała konfiguracja zadań powinna być w pliku Jenkinsfile. Następnie należy skonfigurować zakładkę **Pipeline** w konfiguracji projektu. W polu **Definicja** Musi być wybrane *Pipeline script from SCM*, następnie dalej, wymagany jest adres repozytorium, identyczny jaki jest podany powyżej. Ostatnia rzecz jaka musi być oznaczona, to ścieżka do pliku Jenkinsfile, w którym znajduje się konfiguracja procesu Pipiline, która jest wykorzystywana do odpowiedniej zarządzania procesem w tym przypadku, budowania i testowania. Plik ten jest używany po przechwyceniu zdarzenia żądania POST z poprzedniego punktu i za pomocą niego następuje konfiguracja uruchamianego procesu.



Rys. 1.5. Zakładka ”Pipeline” w konfiguracji projektu w serwisie Jenkins

```
pipeline {
    agent any

    options {
        parallelsAlwaysFailFast()
    }

    stages {
        stage('GIT') {
            steps {
                git '<github_url>'
            }
        }
        stage('DOCKER') {
            steps {
                dir('Grupy/Grupa01/BB401928/Lab04') {
                    sh 'docker-compose up'
```

```
18                 }
19             }
20         }
21         stage("END") {
22             steps {
23                 dir('Grupy/Grupa01/BB401928/Lab04') {
24                     sh 'docker-compose down -v --remove-orphans '
25                 }
26             }
27         }
28     }
29 
30 }
```

Listing 1.1. Przykład pliku Jenkinsfile, wykorzystany we wcześniejszym projekcie

1.2.5. Uruchomienie ekosystemu dockerowego

Po przechwyceniu żądania POST oraz wykorzystaniu konfiguracji znajdującej się w pliku Jenkinsfile, następuje uruchomienie całej procedury.

(a) Początek zadania

(b) Uruchomienie ekosystemu dockera

Rys. 1.6. Przykładowy widok zadania

1.2.6. Pipeline

Aktywności 5-11 oznaczają działanie w związku ze skonfigurowanym plikiem Jsonfile. Proces powinien przebiegać jak na schemacie. Czyli najpierw po uruchomieniu się pipeline, następuje uruchomienie się ekosystemu dockerowego, za pomocą narzędzia docker compose i polecenia:

```
$ docker-compose up
```

Po poprawnym uruchomieniu kontenera build_agent, następuje budowanie aplikacji za pomocą dostępnych w kontenerze narzędzi. W tym przypadku jest to Yarn[1]. Narzędzie to wykorzystuje się m.in. do uruchomienia skonfigurowanych w pliku **package.json** skryptów. Przykładowo:

[1]Menadżer pakietów wykorzystywany zamiast npm do obsługi projektów związanych z językiem javascript. https://yarnpkg.com/

```
$ yarn build:linux
```

możliwe do wykorzystania skrypty w pliku **package.json** są aliasami, które zawierają w sobie często zbiór poleceń służacych do obsługi procesu, np budowania aplikacji. Jeśli budowanie z jakiegokolwiek powodu nie zakończy się sukcesem oraz kontener build_agent się zakończy, przed zbudowaniem aplikacji zwracając inny status exit niż 0, Jenkins zakomunikuje to w postaci błędu na podstronie podglądu projektu w Stage_build. Jeśli zadanie wykona się poprawnie, nastepnym krokiem jest testowanie aplikacji za pomocą kontenera test_agent, ma on wykorzystać zbudowany i sklonowane wcześniej repozytorium, sprawdzić czy istnieje plik o rozszerzeniu ***.deb**, który powinien zostać zbudowany we wczesniejszym etapie. Następnie musi się wykonać dołączony do pliku **package.json** alias *test*

```
$ yarn test
```

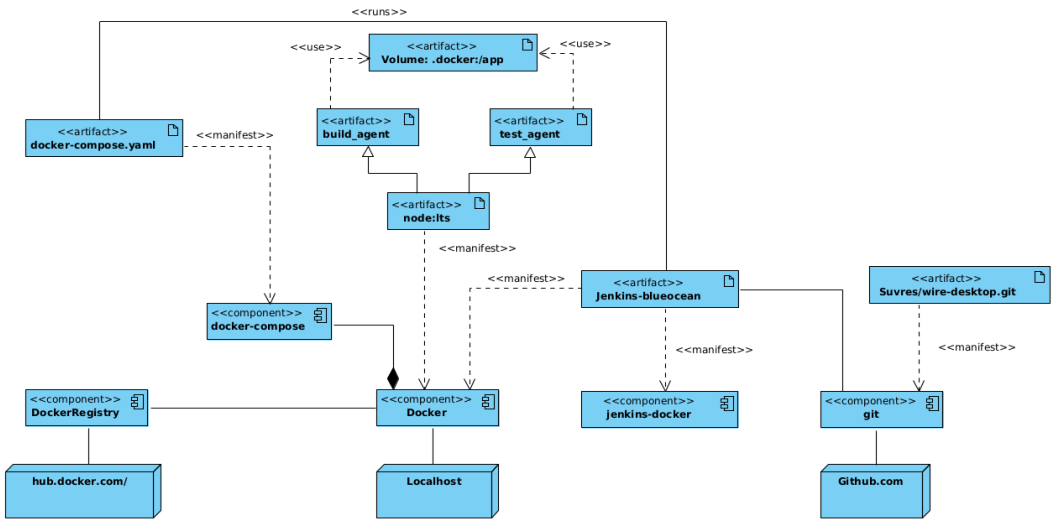
Jeśli wszystkie testy przejdą pomyślnie etap sie zakończy. NAstępnym etapem jest zakończenie pracy dockera za pomocą polecenia:

```
$ docker-compose down -v --remove-orphans
```

1.2.7. Wysłanie zbudowanego pliku deb na serwer

Następnym etap jest wysłanie przygotowanego wcześniej pliku deb na serwer. aby tego dokonać należy w nowym etapie wykorzystać wcześniej przygotowany skrypt **deploy_deb_ssh.sh**, który za pomocą połączenia ssh, wyśle na serwer plik deb, po wcześniejszym sprawdzeniu czy plik istnieje. Po wysłaniu pliku następuje zakończenie etapu, oraz zakończenie całego procesu Pipeline. Po zakończeniu procesu Pipeline w przypadku poprawnie zakończonego procesu, jak i w trakcie niepowodzenia, należy zakończyć cały proces wysłaniem wiadomości email na podany wcześniej adres Project Managera.

1.3. Diagram wdrożeniowy



Rys. 1.7. Diagram aplikacji Suvres/wireapp-desktop

1.4. Podsumowanie

Wykorzystując diagram z **Rys. 1.1** oraz **Rys. 1.7** należy przygotować poszczególne technologie oraz konfiguracje wykorzystywane podczas całego procesu. Linki które mogą być przydatne w stworzeniu powyższej konfiguracji to:

Instalacja podstawowego obrazu Jenkins wykorzystując dockera

https://www.jenkins.io/doc/book/installing/docker/

Github Webhooks

https://docs.github.com/en/github-ae@latest/developers/webhooks-and-events/webhooks

Testowanie WebHooków lokalnie

https://blog.knoldus.com/setting-up-github-webhook-for-localhost/

Konfiguracja plików Jenkinsfile

https://www.jenkins.io/doc/book/pipeline/jenkinsfile/

Spis zdjęć