Plan CI/CD dla wire-desktop

Poniższy plan CI/CD został przygotowany przez Mateusza Stabryłę w dniu 20.05.2021 roku dla aplikacji wire-desktop, komunikatora opartego o system elektron.

Legenda:

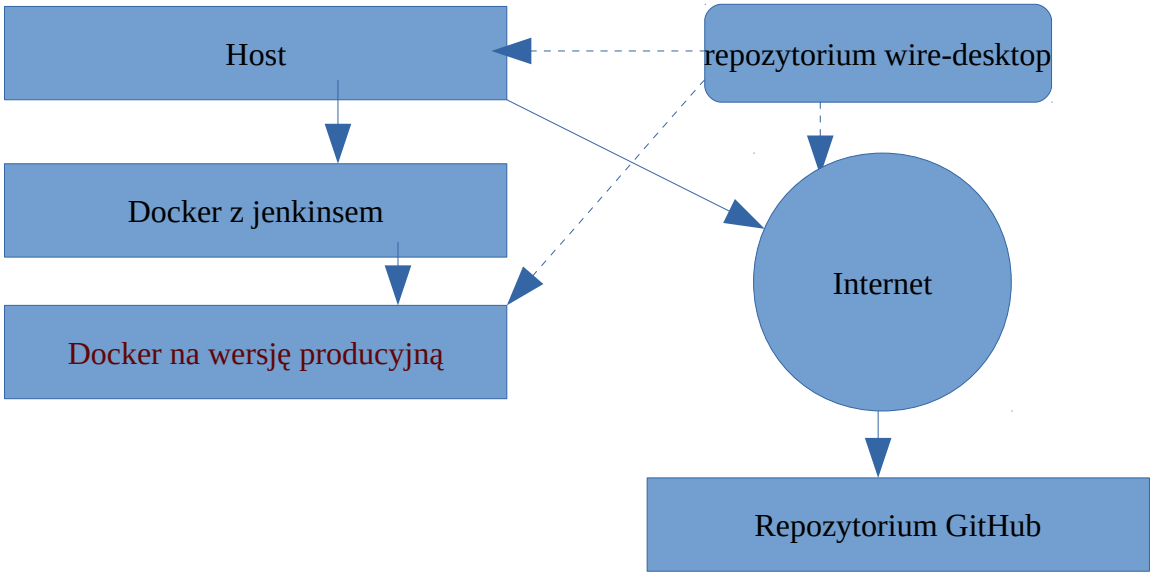
Elementy usunięte z poprzedniego planu | Elementy dodane

Elementy zmienione względem poprzedniego planu

Tabelaryczne zestawienie Activity Diagram

Nazwa kroku	Technologia	Link do pliku konfiguracyjnego	Nr linii	Czy zgodnie z planem
wprowadzenie zmian na gita	git	*brak – komenda w terminalu	-	tak
wykrycie zmian na repozytorium przez Jenkinsa	git hook; curl	[repo]/.git/hooks/Lab09/post-commit.sh	4	tak
Pobranie aktualnej wersji repozytorium	Jenkins	*brak – automatycznie pobieranie przez Jenkinsa	-	tak
Rozpoczęcie budowania aplikacji – stworzenie pliku yarn	npm; yarn;	[repo]/jenkinfile	11	tak
Rozpoczęcie budowania aplikacji – suruchomienie zbudowania	npm; yarn;	[repo]/jenkinfile	12	tak
Rozpoczęcie testów jednostkowych	yarn;	[repo]/jenkinfile	34	tak
Przeniesienie na testowe środowisko produkcyjne	fragment usunięty – nie ma testów integracyjnych			nie
Rozpoczęcie testów integracyjnych	fragment usunięty – nie ma testów integracyjnych			nie
Przygotowanie artefaktów produkcyjnych	yarn;	[repo]/jenkinfile	12	nie – wykonano tę operację wcześniej, nie ma opcji release w kompilacji
Uruchomienie środowiska produkcyjnego	docker	[repo]/jenkinfile	59; 60	nie
Przeniesienie na środowisko produkcyjne	docker	[repo]/jenkinfile	61	nie
Uruchomienie nowej wersji	docker	[repo]/jenkinfile	62	nie
Błąd	jenkins – email extention	[repo]/jenkinfile	23; 47; 74	tak

Poniżej zaprojektowałem Deployment Diagram dla wire-desktop



Tabelaryczne zestawienie Deployment Diagram:

Nazwa elementu	Technologia	Link do pliku konfiguracyjnego	Nr linii	Czy zgodnie z planem
Host	Linux	-	-	tak
repozytorium	git; Github	[repo]/.git/config	7	tak
docker z Jenkinsem	docker; jenkins;	MS306499/ jenkins.sh	3; 5	tak
Docker na wersję produkcyjną	docker;	[repo]/jenkinsfile	59; 60	nie

Słowne omówienie:

1. Na początku istotnym problemem jest wykrycie zmian przez Jenkinsa. Dzięki webowemu API, które on upublicznia możemy to zrobić w bardzo prosty sposób za pomocą zapytać HTTP. Aby zapytania wysyłać w konkretnym momencie, wystarczy do repozytorium naszego programu dołączyć gihooka, najlepiej post-commit, który po każdym kommicie wywoła nowe zadanie w Jenkinsie.
2. Po wywołaniu zmiany należy dokonać aktualizacji repozytorium. Z GitHuba pobieramy aktualną wersję brancha dev za pomocą git pull.
3. Po zaktualizowaniu repo rozpoczynamy proces budowania aplikacji. W naszym przypadku kompilujemy wersję dla Linuxa. Jeżli ta operacja się nie powiedzie to przerywamy Pipeline i powiadamiamy developerów o problemie z budowanie aplikacji. Budowanie odbywa się za pomocą komend yarn oraz yarn build.
4. Następnie przeprowadzamy testowanie. Testy uruchamiamy po wprowadzeniu komendy yarn test. Jeśli ten proces się nie powiedzie to analogicznie przerywamy pipeline i powiadamiamy developerów.
5. Po udanych testach z poziomu jenkinsa uruchamiamy osobnego dockera i rozpoczynamy proces przenoszenia aplikacji na tego dockera. Najlepiej jest korzystać z obrazu posiadającego tylko biblioteki potrzebne do tej aplikacji, najlepiej pobranego z DockerHuba. W kontenerze tymczasowo pobieramy aktualne repozutorium. Jeżli ten proceso się nie powiedzie to powinniśmy powiadomić nie tylko developerów, ale też osoby odpowiedzialne za devopsing, że pojawił się poważny problem.
W tym zadaniu dobrze się może sprawić docker run, oraz Dockerfile z połączonym obrazem buildu i testowania.
6. Po uruchomieniu środowiska integracyjnego uruchamiamy build i testy tego programu, jeśli proces się nie powiedzie to podobnie jak w poprzednim przypadku przerywamy pipeline i powiadamiamy o błędach.
7. Jeśli testy integracyjne się powiodą, to build wygenerowany w Jenkinsie przenosimy na produkcję, czy kolejnego dockera. Tutaj też możemy wykorzystać jakiś inny obraz lub kontener oparty na poprzednim obrazie.

Po uruchomieniu kontenera, przenosimy na niego skompilowaną wersję programu. Jeśli pojawi się jakikolwiek problem z przeniesieniem to powiadamiamy developerów i devops

8. Po przeniesieniu uruchamiamy aplikację za pomocą komendy yarn start. Tutaj też powiadamiamy jak pojawi się jakikolwiek problem.

Omówienie zmian:

Program nie posiada testów integracyjnych, więc część związana z nim usunąłem z pipelina. Tworzenie osobnego dockera dla samego programu (nie wersji produkcyjnej) nie miało sensu, więc ten fragment też został usunięty.