Laboratorium 5 - projekt, Tomasz Ziobrowski

1. Przygorowanie zadania

Przygotowanie pipeline'a oraz SCM

Do pracy wykorzystnao instancję Jenkins działającą jako kontener Dockera przygotowaną podczas poprzedniego zadania.

W panelu Jenkinsa utworzono nowy pipeline, który nazwano "ruru".

Enter	an	itam	nan	ne
cillei	an	пеш	пап	пе

nazwa-

» Required field



Freestyle project

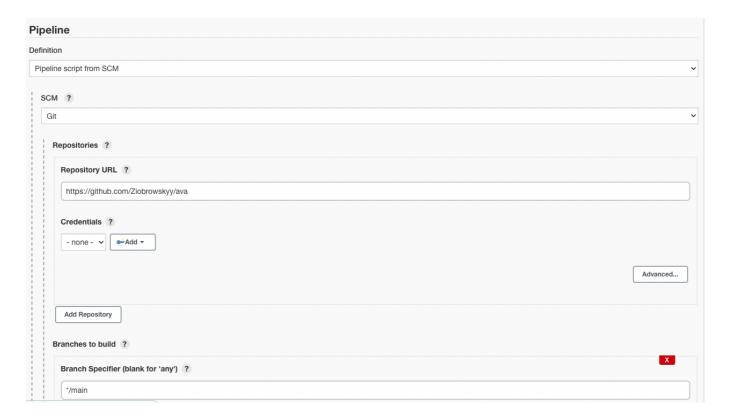
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

W zaawansowanej sekcji ustawień pipeline'a dodano pobieranie skryptu (oraz innych plików) z SCM (Srouce Control Management), który w tym przypadku było sforkowane repozytrium projektu na githubie. Ustawiono również gałąź, z której były zaciągnae zmiany.



Należało również podać ścieżkę do pliku Jenkinsfike, ktory również znajdował się w tym repozytorium.



Zarys Jenkinsfile

W pliku jenkinsfile przygotowano etapy w bloku steps, które zostaną uruchomione podczas budowania rurociagu, oraz w bloku parameters będą znajdowały się parametry builda.

```
🧸 Jenkinsfile
  pipeline {
      agent any
      parameters { …
>
      stages {
           stage("Build") { ...
           stage("Test") { ...
           stage("Publish") { ...
>
```

Parametry builda

Każdy z parametrów budowania ma swój własny typ taki jak boolean, string, choice czy również password. W przypadku tego pipeline'a wybrano jedynie parametry typu bool, oraz string. Były one wykorzystane następnie do ustawień takich akcji jak publikowanie, numer wersji builda (major oraz minor) czy token do poratlu NPM. Każdy z parametrów ma format następujący:

```
<typ_parametru>(name: <nazwa_parametru>, defaultValue: <wartosc_domyślna>, description: <opis_parametru>)
```

Pole description jest opcjonalne i w przypadku nie podania go, zostanie wykorzystany jako label nazwa parametru.

```
parameters {
    booleanParam(name: "PUBLISH", defaultValue: true, description: "Check
to publish build to npm")
    string(name: "VERSION_MAJOR", defaultValue: "1", description: "Major
version of build to be published")
```

```
string(name: "VERSION_MINOR", defaultValue: "0", description: "Minor
version of build to be published")
    string(name: "TOKEN", defaultValue: "", description: "Provide npm
access token, else one in .env file is used")
}
```

Okno wprowadzania parametrów nieznacznie różni się w zależności od tego czy jest wykorzystywany interfejs Blueocean czy Jenkinsa standardowy do odpalenia budowania.

Pipeline ruru
This build requires parameters:
□ PUBLISH Check to publish build to npm
VERSION_MAJOR
1
Major version of build to be published
VERSION_MINOR
0
Minor version of build to be published
TOKEN
Provide npm access token, else one in .env file is used
Build

Input r	equired
Check to pu	ublish build to npm
PUBLIS	БН
Major version	on of build to be published
Minor versi	on of build to be published
0	
Provide npr	m access token, else one in .env file is used
Run	Cancel

Przykładowe wykorzystnaie parametru w kodzie wygląda następująco:

2. Stage "Build"

Celem tego etapu było przygotowanie środowiska programu oraz jego instalacja.

W tym celu wykorzystano plik Dockerfile, który znajdował się również repozytorium projektu a w Jenkinsie pod ścieżką /ava/Dockerfile.

Do projektu wykorzystano multi-stage Dockcerfile, aby móc uniknąć tworzenia wielu plików dockera.

```
FROM node:latest AS build
RUN git clone https://github.com/Ziobrowskyy/ava.git
WORKDIR /ava
RUN npm install

FROM build AS test
WORKDIR /ava
CMD npm run test
```

Jak można zauważyć, jest to bardzo prosty skrypt, który nie wymaga zbyt wiele kroków.

W pliku Jenkinsfile jedynym krokiem było utworzenie obrazu dockera poleceniem docker build — target build —t ava/build:latest ., który przygotowywał obraz gotowego środowiska wraz z zaintalowanym programem. Obraz ten był później wykorzystany do kroku testowania oraz publish.

```
}
```

Działający stage Build

```
Build - 2s
                                                                                             Restart Build 🔀 👤

    Check out from version control

                                                                                                                  <1s
             The recommended git tool is: git
             No credentials specified
             > git rev-parse --resolve-git-dir /var/jenkins_home/workspace/ruru/.git # timeout=10
             Fetching changes from the remote Git repository
              > git config remote.origin.url https://github.com/Ziobrowskyy/ava # timeout=10
             Fetching upstream changes from https://github.com/Ziobrowskyy/ava
              > git --version # timeout=10
             > git --version # 'git version 2.30.2'
             > git fetch --tags --force --progress -- https://github.com/Ziobrowskyy/ava
             +refs/heads/*:refs/remotes/origin/* # timeout=10
              > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
             Checking out Revision 92d35862c5af6cdfff9e59613298a0f1b4588f8d (refs/remotes/origin/main)
             > git config core.sparsecheckout # timeout=10
             > git checkout -f 92d35862c5af6cdfff9e59613298a0f1b4588f8d # timeout=10
             Commit message: "Fix wrong env variable name"
              > git rev-list --no-walk b0f670f67b81ff8fb471ba6d91e019b9e32da78c # timeout=10
                                                                                                                   2s

✓ docker build --target build -t ava/build:latest . — Shell Script

             + docker build --target build -t ava/build:latest .
             Sending build context to Docker daemon 742.9MB
             Step 1/4 : FROM node: latest AS build
              ---> 45aa5693e034
             Step 2/4: RUN git clone https://github.com/Ziobrowskyy/ava.git
              ---> Using cache
             ---> ae7a7bd46167
             Step 3/4 : WORKDIR /ava
             ---> Using cache
              ---> 3b5d448b7726
             Step 4/4 : RUN npm install
              ---> Using cache
              ---> 2080804c70df
             Successfully built 2080804c70df
             Successfully tagged ava/build:latest
```

3. Stage "Test"

Kod Jenkisfile:

```
(...)
    Stage("Test") {
        steps {
            sh "docker build --target test -t ava/test:latest ."
            sh "docker run ava/test:latest"
        }
    }
}
```

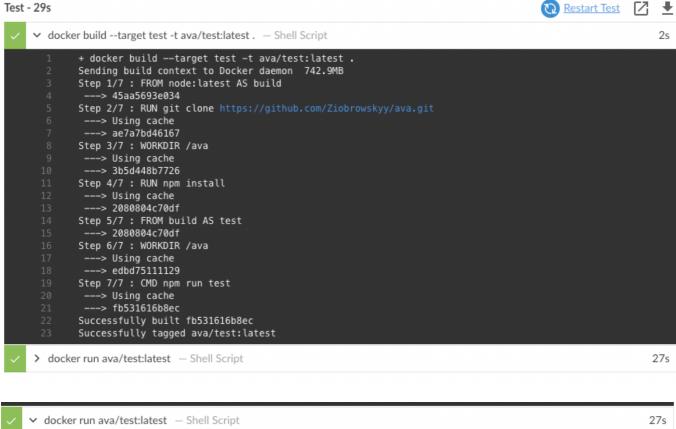
Kod Dockerfile:

```
FROM build AS test
WORKDIR /ava
```

```
CMD npm run test
```

W tym kroku wykorzystany jest obraz z kroku build. W fazie budowania następuje jedynie ustawienie odppowiedniego katalogu roboczego, zaś podoczas docker run następuje wywołanie skryptu testującego aplikację poprzez npm run test.

Działający stage Test



4. Stage "Deploy"

Aplikacja wykorzystana do tego projektu, tj. Ava jest biblioteką, któą wykorzystuje się w projektach npm i służy do testowania aplikacji. Z tego względu nie ma możliwości uruchomienia jej w kroku Deploy, gdyż musiałaby być najpierw zainstalowana i dociągnięta jako dependency do innego projektu, a dopiero potem uruchomiona.

Z tego też względu w pipeline pominięto krok Deploy.

5. Stage "Publish"

Pełny kod Jenkinsfile dla tego kroku:

```
(...)
stage("Publish") {
```

```
when {
                environment(name: "PUBLISH", value: "true")
            }
            agent {
                docker {
                    image "ava/build:latest"
                    args "-u root"
                }
            }
            steps {
                sh "git config user.email thomas.ziobrowski@gmail.com"
                sh "git config user.name Ziobrowskyy"
                script {
                    if(params.TOKEN.equals("")) {
                        echo "Using .env file token"
                        load "$JENKINS HOME/.env"
                        sh "echo
'//registry.npmjs.org/:_authToken=${NPM_TOKEN}' >> ~/.npmrc"
                    } else {
                        echo "Using param token"
                        sh "echo
'//registry.npmjs.org/:_authToken=${params.TOKEN}' >> ~/.npmrc"
                sh "npm version
${params.VERSION_MAJOR}.${params.VERSION_MINOR}.${BUILD_NUMBER}"
                sh "npm publish --access public"
            }
        }
(\ldots)
```

Do warunkowego publiskowania stage wykorzystano parametr pipeline typu boolean PUBLISH oraz klauzulę when.

```
when {
    environment(name: "PUBLISH", value: "true")
}
```

Działa ona w ten sposób, że w momencie gdy jej zwartość zostanie zewaluowana na true, to tylko wtedy dany Stage jest uruchamiany. W tym przypadku sprawdzana jest wartość zmiennej PUBLISH czy jest równa TRUE.

Następnie jako **agent**, czyli środkowisko na rzecz którego będą wykonywane komendy w **bloku steps** wykorzystano kontener dockera utworzony w kroku Build.

```
agent {
    docker {
        image "ava/build:latest"
        args "-u root"
```

```
}
```

```
steps {
    // Zalogowanie się go konta git – potrzebne do kroku npm version
    sh "git config user.email thomas.ziobrowski@gmail.com"
    sh "git config user.name Ziobrowskyy"
    // Sprawdzenie czy token npm został podany jako parametr pipeline
    // orz przygotowanie pliku .npmrc, potrzebnego do autoryzacji npm
publish
    script {
        if(params.TOKEN.equals("")) {
            // Zaciągniecie wartości zmiennej środowiskowej z pliku .env
            echo "Using .env file token"
            load "$JENKINS_HOME/.env"
            sh "echo '//registry.npmjs.org/: authToken=${NPM TOKEN}' >>
~/ npmrc"
        } else {
            // Wykorzystnaie tokenu podanego jako parametr
            echo "Using param token"
            sh "echo '//registry.npmjs.org/:_authToken=${params.TOKEN}' >>
~/.npmrc"
    }
    // Ustawienie wybranej wersji artefaktu
    sh "npm version
${params.VERSION MAJOR}.${params.VERSION MINOR}.${BUILD NUMBER}"
    // Opubliwkowanie programu do npm
    sh "npm publish ——access public"
}
```

Większość kroków, k⁺óre wykonano w kolejnych krokach stage'a po krótce opisano jako komentarze powyżej. Całą procedurę publiskowania do npm mozna podzielić na kilka etapów:

- ustawienie nazwy użytkownika oraz email git (git config user.[email|name] <wartość>), które są potrzebene do nastpęngo ustawienia wersji aplickacji komendą npm version
- przygotowanie pliku npmrc, który jest wymagany do autoryzacji uzytkownika w npm. W ten sposób wiadomo, kto jest autorem danego repo do opubliskowania. W celu autoryzacji wykorzystywany jest token npm.
 - aby wygenrować token, należy wejść w ustaienia konta na stronie (npm)[npmjs.com], a następnie wejść w zakładkę Aceess tokens. Wyświetlają się tam wygenrowane tokeny, oraz

opcja utworzenia nowego

Access Tokens Generate New Token Delete Selected Tokens Type Name Created Delete npm_6kz4.....kNlK Publish jenkins 20 hours ago npm_0foT.....7kuu Publish 20 hours ago

 przy generowaniu tokenu należy podać nazwę, która umożliwi późniejszą jego identyfikację oraz wybrać prawa, które będą przyznae po zalogowaniu z wykorzytaniem tegż oto tokenu. W naszym przypadku należy wybrać opcję publish, ponieważ będziemy chcieli opubliskować z pipeline'a nasz projekt do npm.

New Access Token

Access tokens can be used instead of your password when using the npm CLI to download or publish packages.

Name

Select type

The type of access token defines its permissions. Read more about types of access tokens.

Read-only
 A read-only token can download public or private packages from the npm registry.

 Automation
 An automation token will bypass two-factor authentication when publishing. If you have two-factor authentication enabled, you will not be prompted when using an automation

two-factor authentication enabled, you will not be prompted when using an automation token, making it suitable for CI/CD workflows.

Publish

A publish token can read **and** publish packages to the npm registry. If you have <u>two-factor authentication</u> (2FA) enabled, it **will** be required when using this token.

Generate Token

- zakładając, że token jest dos⁺epny jako zmienna należy przygotować plik _npmrc komendą echo
 '//registry.npmjs.org/:_authToken=\${NPM_TOKEN}' >> ~/.npmrc, która umieści adres
 registry npm oraz token autoryzacyjny w tym pliku
 - Oprócz przekazania tokenu jako paramtetr pipeline' jest możliwość użycia pliku env w katalogu głównym Jenkinsa. Przygotowujemy w następujący sposób plik:

```
ava > 🌣 .env
       NPM_TOKEN="<token_npm>"
```

o następnie należy go wysłać do kontenera Jenkinsa

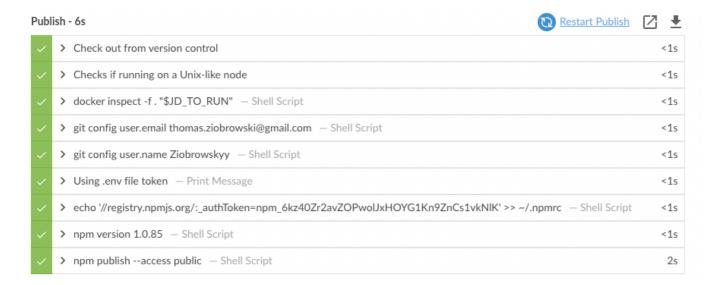
```
anon@Tomaszs-MacBook-Air-2 ava % file .env
.env: ASCII text, with no line terminators
nonTomaszs-MacBook-Air-2 ava % docker ps
CONTAINER ID IMAGE
NAMES
 NAMES
4b636997315 docker:dind
                                                              "dockerd-entrypoint..." 20 hours ago
                                                                                                                    Up 20 hours
                                                                                                                                       2375/tcp, 0.0.0.0:2376->2376/tcp
jenkins-docker
4d68510e8f4 myjenkins-blueocean:2.332.3-1 "/sbin/tini -- /usr/-" 20 hours ago Up 20 hours
jenkins-blueocean
                                                                                                                                      0.0.0.0:8080->8080/tcp, 0.0.0:50000->50000/tcp
      Tomaszs-MacBook-Air-2 ava % docker cp .env a4d:/var/jenkins_home/
```

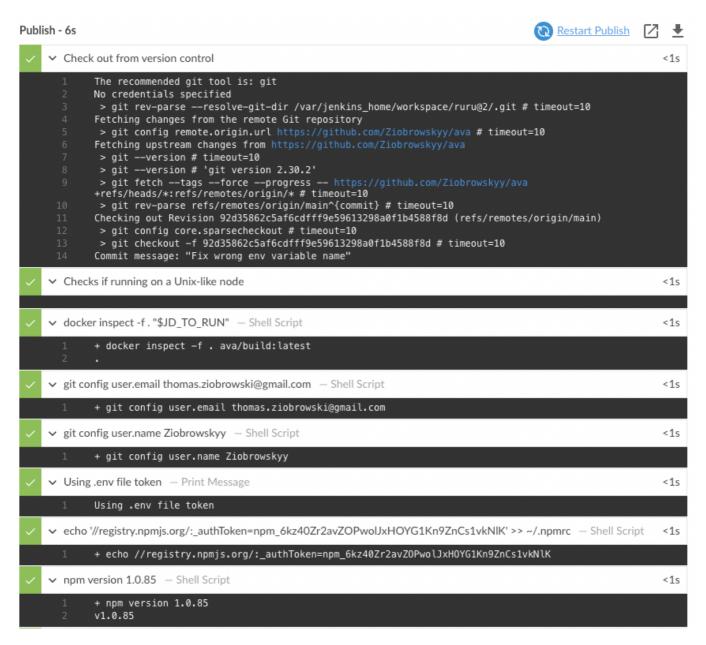
o można go odczytać z pliku Jenkinsfile wykorzystując komendę load <sciezka_do_pliku>, a nastepnie wykorzystać jako zmienną środowiskową jak tu:

```
load "$JENKINS_HOME/.env" //zawartość pliku .env: NPM_TOKEN="<token>"
sh "echo '//registry.npmjs.org/: authToken=${NPM TOKEN}' >> ~/.npmrc"
```

- kolejnym krokiem będzie ustawienie odpowiedniej wersji programu do publikacji wykorsztując komende npm version <nr_wersji>, gdzie numer wersji musi zawierać 3 cyrfy oddzielone kropkami w formacie <MAJOR> . <MINOR> . <FIX>.
 - o numer wersji major oraz minor pobrano jako paramtetr pipeline'a a jako ostatnią wartość użyto numer builda Jenkinsa jako npm version \${params.MAJOR}.\${params.MINOR}. {BUILD NUMBER}
- opublikowanie projektu do repozytorium npm komendą npm publish —access public
 - zakładając że token npm podany wcześniej jest prawidłowy oraz nastąpiłą porawnie autoryzacjia użytkownika krok ten powinien przebiec prawidlowo
 - o numer wersji opubliskowanego progrmau nie moze się różnić, ponieważ inaczej może nastąpic blad
 - o posiadając jedynie darmowe konto na npm należy publiskować projekty jako publiczne z flagą --access public, inaczej wystąpi bład w tym kroku

Działający stage Publish





```
2s

    npm publish --access public — Shell Script

        + npm publish --access public
        npm notice
        npm notice package: @ziobrowskyy/ava@1.0.85
        npm notice === Tarball Contents ===
       npm notice 61B entrypoints/cli.mjs
       npm notice 3.1kB entrypoints/eslint-plugin-helper.cjs
       npm notice 66B entrypoints/main.cjs
npm notice 48B entrypoints/main.mjs
       npm notice 68B
                           entrypoints/plugin.cjs
       npm notice 123B
                         entrypoints/plugin.mjs
       npm notice 386B
                           index.d.ts
       npm notice 10.2kB lib/api.js
       npm notice 24.8kB lib/assert.js
        npm notice 351B
                           lib/chalk.js
       npm notice 14.5kB lib/cli.js
       npm notice 1.3kB lib/code-excerpt.js
```

```
npm notice 1.5kB types/try-fn.d.ts
npm notice === Tarball Details ==
npm notice name:
                        @ziobrowskyy/ava
npm notice version:
                         1.0.85
npm notice filename:
                         @ziobrowskyy/ava-1.0.85.tgz
npm notice package size: 68.2 kB
npm notice unpacked size: 261.7 kB
                         66b78936c541077acdd5e5f15064eecf2451ec21
npm notice shasum:
npm notice integrity:
                          sha512-J86esJG14EmZP[...]aSjQlOy7ekmpg==
npm notice total files: 66
npm notice
npm notice Publishing to https://registry.npmjs.org/
+ @ziobrowskyy/ava@1.0.85
```

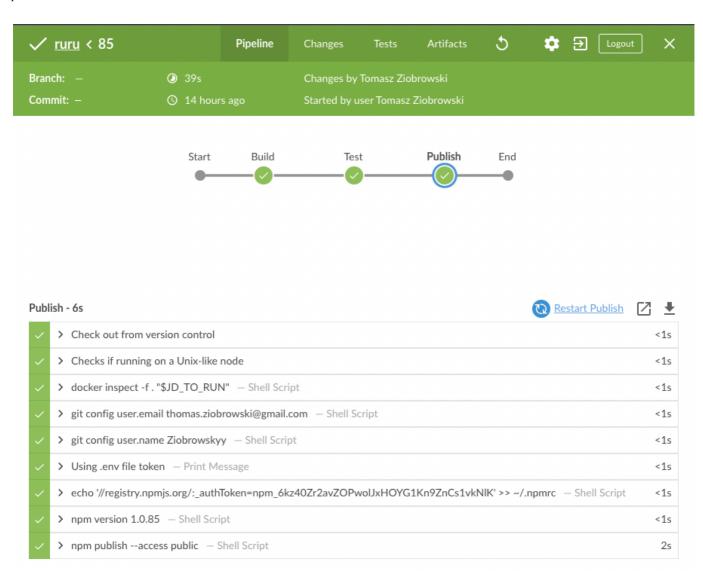
Dzialajacy build pipeline'a

Gotowy plik Jenkinsfile:

```
pipeline {
    agent any
        booleanParam(name: "PUBLISH", defaultValue: true, description:
"Check to publish build to npm")
        string(name: "VERSION_MAJOR", defaultValue: "1", description:
"Major version of build to be published")
        string(name: "VERSION_MINOR", defaultValue: "0", description:
"Minor version of build to be published")
        string(name: "TOKEN", defaultValue: "", description: "Provide npm
access token, else one in .env file is used")
    stages {
        stage("Build") {
            steps {
                sh "docker build -- target build -t ava/build:latest ."
        }
        stage("Test") {
            steps {
                sh "docker build --target test -t ava/test:latest ."
                sh "docker run ava/test:latest"
            }
        }
```

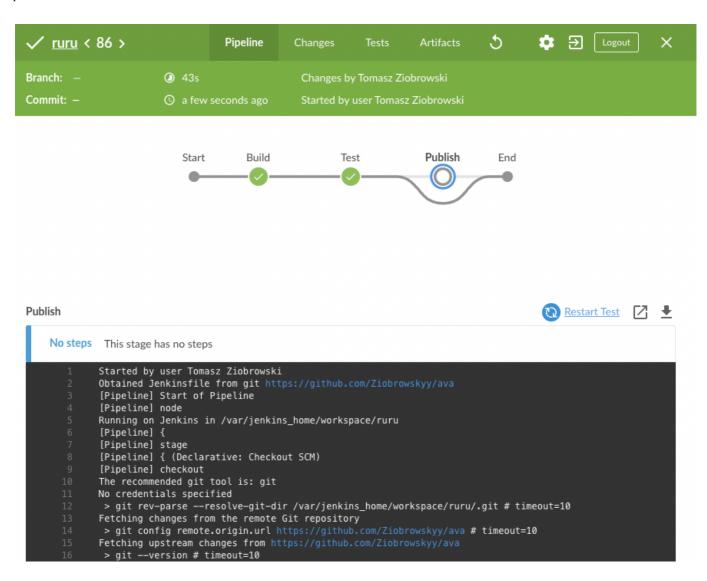
```
stage("Publish") {
            when {
                environment(name: "PUBLISH", value: "true")
            }
            agent {
                docker {
                    image "ava/build:latest"
                    args "-u root"
                }
            }
            steps {
                sh "git config user.email thomas.ziobrowski@gmail.com"
                sh "git config user.name Ziobrowskyy"
                script {
                    if(params.TOKEN.equals("")) {
                        echo "Using .env file token"
                        load "$JENKINS HOME/.env"
                        sh "echo
'//registry.npmjs.org/:_authToken=${NPM_TOKEN}' >> ~/.npmrc"
                    } else {
                        echo "Using param token"
                        sh "echo
'//registry.npmjs.org/:_authToken=${params.TOKEN}' >> ~/.npmrc"
                sh "npm version
${params.VERSION_MAJOR}.${params.VERSION_MINOR}.${BUILD_NUMBER}"
                sh "npm publish --access public"
            }
        }
   }
}
```

• wraz z publish



• bez publish

Input re	equired
Check to pu	blish build to npm
PUBLIS	Н
Major versio	on of build to be published
1	
Minor version	on of build to be published
0	
Provide npm	n access token, else one in .env file is used
D	Const
Run	Cancel



6. Napotkane problemy

- Jednym z natrętnie mnie spotykających problemów podczas rezalizacji tego projektu była konieczność pamietania, cyz podczas uruchomienie jenkisa, działa również kontener dina Dockera. Errory, które się w przeciwnym wypadku pojawiały były bardzo niepomocne i sprawiały wiele problemów, ponieważ ciężko było je wyszukać w internecie :^(Dopiero w momencie poddanaia się i reinsatacji Jenkisa na nowo odkryłem, że własnie dockerowy dind był źródłem kłopotów i frustracji.
 - o zmarnowany czas: 4-5h
- W przypadku Jenkisa i paramterów budowania, od roku 2018 nie potrafią naprawić aktualizacji tychże paramterów. Kilkukrotne odpalanie pipeline'a nie rozwiązywało problemu z cachowaniem poprzednich parametów. Tak samo uruchamianie ponowne Jenkisa nie przyniosło spodziewanych rezultatów. Rozwiązanie było takie, że należało w interfejsu Blueocean przejść do standardowego, tam odpalić jescze raz build (a czaem nie trzeba było) i magicznie pojawiały się już zaktualizaowane paramery (???). Po odpaleniu builda ze standardowego UI i przejściu z powrotem na Blueocean tam również były one zaktualizowane.
 - o zmarnowany czas: 2h