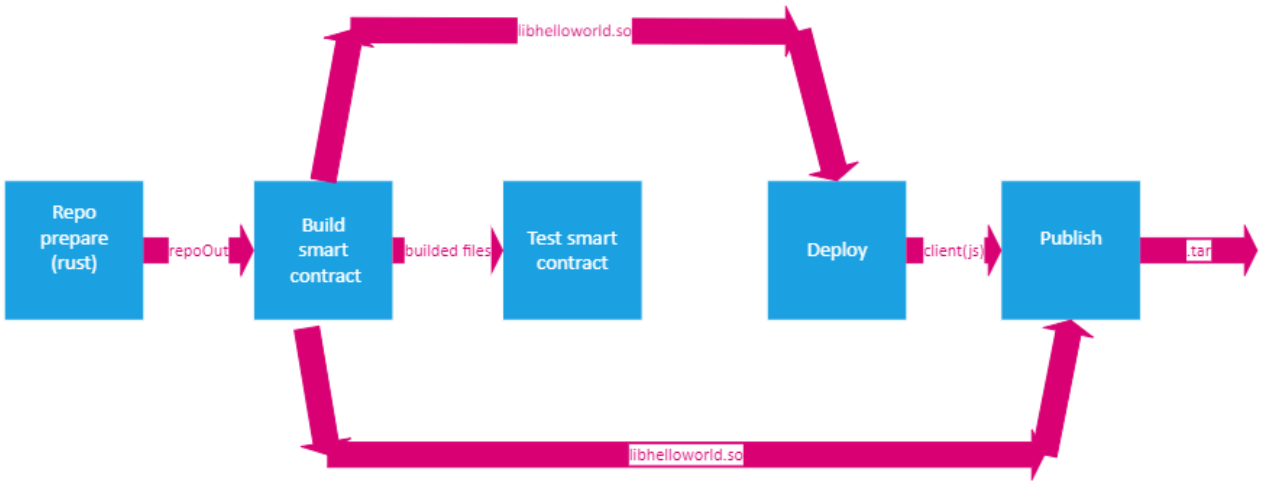
# Sprawozdanie


## Maciej Cholewa


## Wstęp

Celem sprawozdania jest pokazanie budowania pipelinu krok po kroku (wraz z napotkanymi problemami). Spodziewany efekt końcowy działania pipelinu:




1. Pierwszym etapem pipelinu jest sklonowanie repozytorium na volumen wyjściowy, który bedzie wejściem do kolejnego kontenera. Obrazem bazowym jest **rust:latest**, gdyż będzie on używany w kolejnych etapach i ma on zainstalowanego gita.

```
stage('Repo prepare') {
    steps {
        sh 'docker volume create repoOut'
        sh '''
        docker run  --mount source=repoOut,destination=/repoOut rust:latest sh -c
        'cd "repoOut/" && git clone "https://github.com/solana-labs/example-helloworld.git" && ls'
        '''
    }
}
```

```
stage('Repo prepare') {
    steps {
        sh 'docker volume create repoOut'
        sh '''
        docker run  --mount source=repoOut,destination=/repoOut rust:latest sh -c
        'cd "repoOut/" && git clone "https://github.com/solana-labs/example-
helloworld.git" && ls'
        '''
        }
    }
```

Problem napotkany przy tym kroku to:

```
WorkflowScript: 8: unexpected token: && @ line 8, column 113.
   n rust sh -c 'cd dyskietkaIn/ && git clo
                                  ^
```

Udało się go rozwiązać przez użycie ''' ''' do przekazywania poleceń i " " do przkazywania argumentów w pipelinie

2. Kolejnym napotkanym problemem, było zostawanie utworzonych volumenów, po uruchomieniu pipelinum kolejny raz.

```
+ docker run --mount source=repoOut,destination=/repoOut rust sh -c cd "repoOut/" && git clone "https://github.com/solana-labs/example-helloworld.git" && ls
fatal: destination path 'example-helloworld' already exists and is not an empty directory.
```

Rozwiązałem ten problem poprzez dodanie sekcji **post**, w której usuwam wszystkie kontenery oraz volumeny, utworzone podczas trwania pipelinu.

```
post{
    always{
          sh '''docker rm -f $(docker ps -a -q)'''
          sh '''docker volume rm $(docker volume ls -q)'''
    }
}
```

3. Kolejnym krokiem było dodanie kroku **Build smart contract**, w którym obrazem bazowym jest wyżej wspomniany rust:latest - posiada on wszystkie dependencje potrzebne do zbudowania i testowania programu. W tym etapie kopiuje z volumenu wejściowego **program-rust** do volumenu wyjściowego, w którym buduje program.

```
 stage('Build smart contract') {
     steps {
         sh 'docker volume create buildedSmartContract'
         sh 'docker volume create smartContract'
         sh '''
         docker run  --mount source=smartContract,destination=/smartContract
--mount source=repoOut,destination=/repoIn
         --mount
source=buildedSmartContract,destination=/buildedSmartContract rust:latest
sh -c
         'cp -r "repoIn/example-helloworld/src/program-rust"
"/buildedSmartContract"  && cd "/buildedSmartContract/program-rust" &&
         cargo build  && ls && cp -r "/buildedSmartContract/program-
rust/target/debug/libhelloworld.so" "/smartContract"'
         '''
     }
 }
```

```
stage('Build smart contract') {
    steps {
        sh 'docker volume create buildedSmartContract'
        sh 'docker volume create smartContract'
        sh '''
        docker run  --mount source=smartContract,destination=/smartContract --mount source=repoOut,destination=/repoIn
        --mount source=buildedSmartContract,destination=/buildedSmartContract rust:latest  sh -c
        'cp -r "repoIn/example-helloworld/src/program-rust"  "/buildedSmartContract"  && cd "/buildedSmartContract/program-rust" &&
        cargo build  && ls && cp -r "/buildedSmartContract/program-rust/target/debug/libhelloworld.so" "/smartContract"'
        '''
    }
}
```

4. Kolejnym krokiem pipelinu jest **Test smart contract**, w którym wykonujemy testy na zbudowanym wczesniej programie

```
stage('Test smart contract') {
    steps {
        sh '''
        docker run --mount
source=buildedSmartContract,destination=/buildedSmartContract rust:latest
        sh -c ' cd "/buildedSmartContract/program-rust" && cargo test'
        '''
    }
}
```

```
stage('Test smart contract') {
    steps {
        sh '''
        docker run --mount source=buildedSmartContract,destination=/buildedSmartContract rust:latest
        sh -c ' cd "/buildedSmartContract/program-rust" && cargo test'
        '''
    }
}
```

5. Kolejnym krokiem pipelinu jest **Deploy**, w którym obrazem bazowym jest node, gdyż muszę zbudować klienta przed uruchomieniem go, który jest napisany w js. Jednak najpierw zainstalowałem oraz uruchomiłem klaster solany, aby móc tam zamieścić program.

Pierwszym napotkanym problemem był błąd podczas generowania kluczy przez solanę, podczas instalacji i przygotowania solany do działania.

```
Generating a new keypair
thread 'main' panicked at 'called `Result::unwrap()` on an `Err` value: Os { code: 6, kind: Uncategorized, message: "No such device or address" }', keygen/src/keygen.rs:576:92
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
```

Udało mi się naprawić problem poprzez zainstalowanie wersji **stable**, a nie **v1.10.8**, która była podana w dokumentacji.

Następnym problemem, na tym etapie był problem z wrzuceniem programu na solanę. Aby to zrobić należy uruchomić klaster:

```
solana-test-validator
```

```
+ docker run --name solanaCon -d --mount source=smartContract,destination=/smartContract node:latest sh -c sh -c "$(curl -sSfL https://release.solana.com/stable/install)" && export
PATH="/root/.local/share/solana/install/active_release/bin:$PATH"  && echo "solana dziala" && solana --version  && pwd && solana config set --url "localhost" && solana-keygen "new" && solan
validator
03696f3c859b88be8abf47b7eeb3946f3947f8ff1210ab0ecc996b28d4e402b8
+ sleep 90
+ docker logs solanaCon
downloading stable installer
  stable commit 0fd7935 initialized
Adding
export PATH="/root/.local/share/solana/install/active_release/bin:$PATH" to /root/.profile

Close and reopen your terminal to apply the PATH changes or run the following in your existing shell:

export PATH="/root/.local/share/solana/install/active_release/bin:$PATH"

solana dziala
solana-cli 1.9.22 (src:0fd79356; feat:2945818700)
/
Config File: /root/.config/solana/cli/config.yml
RPC URL: http://localhost:8899
WebSocket URL: ws://localhost:8900/ (computed)
Keypair Path: /root/.config/solana/id.json
Commitment: confirmed
Generating a new keypair

Ledger location: test-ledger
Log: test-ledger/validator.log
Initializing...
Waiting for fees to stabilize 1...
Waiting for fees to stabilize 2...
Connecting...
Identity: ApSrxHf922RpA3d3jjK2322sqskYNN4qSKBAaU1vXqna
Genesis Hash: FZAiXN6rW8Ty2LZBymYhBM5oGPM92Gy56gX2rwN5chfW
Version: 1.9.22
Shred Version: 21841
Gossip Address: 127.0.0.1:1024
TPU Address: 127.0.0.1:1027
JSON RPC URL: http://127.0.0.1:8899
00:00:02 | Processed Slot: 1 | Confirmed Slot: 1 | Finalized Slot: 0 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 1 | ◎500.000000000
00:00:02 | Processed Slot: 2 | Confirmed Slot: 2 | Finalized Slot: 0 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 2 | ◎499.999995000
00:00:02 | Processed Slot: 3 | Confirmed Slot: 2 | Finalized Slot: 0 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 3 | ◎499.999995000
00:00:03 | Processed Slot: 3 | Confirmed Slot: 3 | Finalized Slot: 0 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 3 | ◎499.999990000
00:00:03 | Processed Slot: 4 | Confirmed Slot: 4 | Finalized Slot: 0 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 4 | ◎499.999985000
00:00:03 | Processed Slot: 4 | Confirmed Slot: 4 | Finalized Slot: 0 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 4 | ◎499.999985000
00:00:03 | Processed Slot: 5 | Confirmed Slot: 5 | Finalized Slot: 0 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 5 | ◎499.999980000
00:00:04 | Processed Slot: 6 | Confirmed Slot: 6 | Finalized Slot: 0 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 6 | ◎499.999975000
00:00:04 | Processed Slot: 6 | Confirmed Slot: 6 | Finalized Slot: 0 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 6 | ◎499.999975000
00:00:04 | Processed Slot: 7 | Confirmed Slot: 7 | Finalized Slot: 0 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 7 | ◎499.999970000
00:00:04 | Processed Slot: 7 | Confirmed Slot: 7 | Finalized Slot: 0 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 7 | ◎499.999970000
```

Udało mi się urchomić klaster w kontenerze z flagą -d, co pozwala na dostpę do kontenera za pomocą **docker exec**. Logi ze zdjęcia są zebrane za pomocą **docker logs solanaCon**. Niestety na kolejnym etapie nie udało się zamieścić programu.

```
00:00:32 | Processed Slot: 75 | Confirmed Slot: 75 | Finalized Slot: 43 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 75 | ◎499.999630000
00:00:33 | Processed Slot: 76 | Confirmed Slot: 76 | Finalized Slot: 44 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 76 | ◎499.999625000
00:00:33 | Processed Slot: 77 | Confirmed Slot: 77 | Finalized Slot: 45 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 77 | ◎499.999620000
00:00:33 | Processed Slot: 77 | Confirmed Slot: 77 | Finalized Slot: 45 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 77 | ◎499.999620000
00:00:33 | Processed Slot: 78 | Confirmed Slot: 78 | Finalized Slot: 46 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 78 | ◎499.999615000
00:00:34 | Processed Slot: 78 | Confirmed Slot: 78 | Finalized Slot: 46 | Full Snapshot Slot: - | Incremental Snapshot Slot: - | Transactions: 78 | ◎499.999615000
+ docker exec solanaCon sh -c cd smartContract  && solana program deploy "libhelloworld.so"
sh: 1: solana: not found
```

Przy próbie wrzucenia programu, **sh** nie widzi zainstalowanej solany, co uniemożliwia zamieszczenie porgramu na sieć solany.

```
    stage('Deploy') {
        steps {
            sh '''
            docker run --name solanaCon -d --mount
source=smartContract,destination=/smartContract  node:latest
            sh -c 'sh -c "$(curl -sSfL https://release.solana.com/stable/install)"
&&
            export
PATH="/root/.local/share/solana/install/active_release/bin:$PATH"  &&
            echo "solana dziala" && solana --version  && pwd && solana config set
--url "localhost" && solana-keygen "new" && solana-test-validator'
            sleep 90
```

```
            docker logs solanaCon
            docker exec solanaCon sh -c 'solana program deploy "smartContract
/libhelloworld.so"'
            '''
        }
    }
```

```
stage('Deploy') {
    steps {
        sh '''
        docker run --name solanaCon -d --mount source=smartContract,destination=/smartContract  node:latest
        sh -c 'sh -c "$(curl -sSfL https://release.solana.com/stable/install)" &&
        export PATH="/root/.local/share/solana/install/active_release/bin:$PATH"  &&
        echo "solana dziala" && solana --version  && pwd && solana config set --url "localhost" && solana-keygen "new" && solana-test-validator'
        sleep 90
        docker logs solanaCon
        docker exec solanaCon sh -c 'solana program deploy "smartContract /libhelloworld.so"'
        '''
    }
```

6. Ostatnim etapem jest **Publish**, w tym etapie **smart contract** (plik z rozszerzeniem .so), client napisany w js wraz z instrukcją jest pakowany do paczki.tar.gz. Program nie zawiera instalatora, ponieważ potrzebuje być wgrany na klaster solany, gdzie zostajnie uruchomiony (nie potrzeba instalatora), a gdy uruchomimy klienta to zostanie nam pokazany efekt działania programu.

Etap ten bazuje na parametrze **PROMOTE**, jeżeli jego wartość jest true to nowa wersja zostanie opublikowana.

```
parameters {
    booleanParam(name: 'PROMOTE', defaultValue: true, description: '')
}



stage('Publish'){
    when{
        expression { params.PROMOTE ==~ /(?i)(Y|YES|T|TRUE|ON|RUN)/ }
    }
    steps{
        sh'''
        docker volume create  finalVersion
        docker run --mount source=smartContract,destination=/smartContract --mount
source=finalVersion,destination=/finalVersion --mount
source=repoOut,destination=/repoOut rust:latest sh -c 'cp -r
"/smartContract/libhelloworld.so" "/finalVersion" cp -r "repoOut/example-
helloworld/src/client/" "/finalVersion" && cd "finalVersion" && touch
"instructions.txt" && echo "For deploying program to Solana network follow the
instuctions https://github.com/solana-labs/example-
helloworld/blob/master/README.md" >> "instructions.txt" && tar -zcvf sol.tar.gz
"/finalVersion"'
        '''
        }
    }
```

```
stage('Publish'){
    when{
        expression { params.PROMOTE ==~ /(?i)(Y|YES|T|TRUE|ON|RUN)/ }

    }
    steps{
        sh'''
        docker volume create  finalVersion
        docker run --mount source=smartContract,destination=/smartContract --mount source=finalVersion,destination=/finalVersion --mount source
        '''
    }
```

7. Efekt końcowy działania pipelinu, z wykliczeniem etapu **deploy**:

Recent Changes

## Stage View

| | Repo prepare | Build smart contract | Test smart contract | Deploy | Publish | Declarative: Post Actions |
|---|---|---|---|---|---|---|
| Average stage times: (Average full run time: ~3min 31s) | 2s | 58s | 1min 6s | 233ms | 767ms | 1s |
| #122 May 16 16:21 No Changes | 3s | 1min 21s | 2min 2s | 409ms | 1s | 1s |
| #121 May 16 16:20 No Changes | | | | | | |