Opis skonteneryzowanej wersji rdzenia sieci 5G

[**1. Początkowe założenia 2**](#_awh9xk4xk0ar)

[1.1 Używane skróty 2](#_cfrxmon66cs3)

[1.2 Wybrana implementacja rdzenia sieci 5G 2](#_p9nh8b3vc53g)

[1.3 Schemat połączeń 2](#_anhvp44wai8v)

[**2. Konfiguracja rdzenia 3**](#_tjuy8aphk9lg)

[2.1 Obrazy kontenerów (Docker) 3](#_lwq77msdwrrm)

[2.2 Kubernetes 3](#_53d6gyxu1xr7)

[**3. Skrypty automatyzacyjne 4**](#_vbuxqt2momkd)

[3.1 Założenia 4](#_opgft95b0ofk)

[3.2 Przykładowy skrypt 4](#_flj3ulapu265)

[**4. Podsumowanie 5**](#_vtzyoxhmij7z)

# 

# Początkowe założenia

## 1.1 Używane skróty

Data Network (DN) - sieć zewnętrzna

Core - rdzeń Open5GS

NF - funkcja rdzenia

gNB - stacja bazowa (gNodeB)

UE - terminal końcowy

## 1.2 Wybrana implementacja rdzenia sieci 5G

Wybrana implementacja rdzenia sieci 5G: Open5GS (<https://open5gs.org/>). Użyto wersji 5G SA (standalone).

## 1.3 Schemat połączeń

# Konfiguracja rdzenia

## 2.1 Obrazy kontenerów (Docker)

**Baza -** oparty na Ubuntu 20.04. Zawiera wymagane biblioteki i pliki wykonywalne. Wykorzystany jako podstawa obrazów kontenerów poszczególnych elementów rdzenia.

**Builder -** oparty na Baza. Zawiera skompilowany rdzeń (z forka repozytorium – <https://github.com/KedArch/open5gs>**)**

**Obrazy elementów rdzenia**- oparte na Baza

- zawiera implementacje Open5GS dla odpowiedniego NF (pliki wykonywalne oraz biblioteki z Buildera)  
- standardowo uruchamiane skryptem setup.sh ze ścieżki /root/setup.sh  
- konfiguracja oparta na zmiennych środowiskowych

## 2.2 Kubernetes

kube-apiserver jest zmodyfikowany, by pozwolić na użycie portów 2152 oraz 38412 w Serwisach typu NodePort, wykorzystywanych w UPF oraz AMF.

Komunikacja pomiędzy Podami jest zrealizowana za pomocą serwisów ClusterIP, natomiast pomiędzy Podem a zewnętrzną maszyną za pomocą serwisu NodePort. Pody komunikują się wewnętrznie za pomocą etykiet (nazw serwisów), np. nrf-service.

Przy inicjalizacji bazy danych można uruchomić osobny Pod (mongo-add-admin), który dodaje do niej konto administratora wykorzystywane w WebUI. Baza danych, w przeciwieństwie do elementów rdzenia, to StatefulSet, ponieważ ułatwia to dodanie *Persistent Volume*.

Pody są tworzone na podstawie obrazów opisanych wyżej przy pomocy tagów.

**Pliki YAML elemtów rdzenia (Deploymenty)**- zawierają odpowiedni Deployment (lub w przypadku MongoDB StatefulSet) oraz powiązane Serwisy  
- konfigurują kontenery przy pomocy zmiennych środowiskowych lub nazw kontenerów (w przypadku AMF) czy IP na interfejsie kontenera (w przypadku UPF, omija to problem rozgłaszania adresu w NRF).

# 

# Skrypty automatyzacyjne

## 3.1 Założenia

Skrypty są automatycznie uruchamiane podczas tworzenia odpowiedniego kontenera. Tworzą one plik konfiguracyjny w kontenerze oraz uruchamiają konkretne NF.

## 3.2 Przykładowy skrypt

Wszystkie NF są konfigurowane i uruchamiane przez skrypt /root/setup.sh, który przygotowuje je do pracy. Przykładowy skrypt do konfiguracji SCP.

#!/bin/sh

CONFIG=/etc/open5gs/scp.yaml

# przeniesienie oryginalnej konfiguracji, jeśli istnieje

if ! [ -f "$CONFIG-original" ]; then

mv $CONFIG $CONFIG-original

fi

# ustawienie parametrów kontenera na domyślne lub zgodnie z podanymi zmiennymi środowiskowymi

if [ -z "$DB\_URI" ]; then

DB\_URI=mongodb://localhost/open5gs

fi

if [ -z "$SCP\_ADDR" ]; then

SCP\_ADDR=127.0.1.10

fi

if [ -z "$SCP\_PORT" ]; then

SCP\_PORT=7777

fi

if [ -z "$NRF\_ADDR" ]; then

NRF\_ADDR="127.0.0.10 ::1"

fi

# jeśli jest więcej niż jedna możliwa wartość w zmiennej, formatuje zawartość tej zmienne zgodnie ze składnią pliku konfiguracyjnego

NRF\_ADDR=`echo $NRF\_ADDR | sed 's/ /\n - /g'`

if [ -z "$NRF\_PORT" ]; then

NRF\_PORT=7777

fi

# plik konfiguracyjny jest zapisywany w miejsce, które odczytuje plik wykonywalny NF

printf "db\_uri: $DB\_URI

logger:

file: /var/log/open5gs/scp.log

scp:

sbi:

- addr: $SCP\_ADDR

port: $SCP\_PORT

nrf:

sbi:

- addr:

- $NRF\_ADDR

port: $NRF\_PORT

parameter:

max:

time:

" > $CONFIG

/bin/open5gs-scpd

# Podsumowanie

Rezultatem działania dostarczonych plików jest działająca wersja rdzenia sieci 5G opartego na Open5GS. Wykonana implementacja pozwala na transfer danych od wielu UE (subskrybentów) do podłączonego DN oraz odporność na awarie NF (o ile nie zostały zmodyfikowane).

Nie zostało sprawdzone działanie połączeń IPv6, podział subskrybentów na strefy, działanie wielu AMF oraz UPF jednocześnie. Funkcje te zostaną wprowadzone po przejściu testów.

Na ten moment nie są wspierane połączenia TLS pomiędzy NF oraz replikacja bazy danych.