Akademia Górniczo-Hutnicza

Dokumentacja do projektu

TIG - system kontroli wersji

z przedmiotu

Programowanie Sieciowe

Elektronika i Telekomunikacja III

Bartłomiej Osiak

Piątek 16:45

prowadzący: dr inż. Janusz Gozdecki

30.12.2025

Opis działania systemu TIG

System TIG został zaprojektowany w oparciu o protokół sieciowy IPv6 oraz protokół transportowy TCP, które zapewniają niezawodną komunikację pomiędzy klientem a serwerem. Klient systemu ma możliwość wysyłania oraz pobierania całych katalogów do lub z bieżącej ścieżki roboczej w swoim systemie plików.

Serwer gromadzi dane we wcześniej zdefiniowanej strukturze katalogów pod ścieżką **/TIG/srv/data**. Informacje pozyskiwane od klienta podzielone są na cztery kategorie: **commits**, **repos**, **backups oraz list**.

W katalogu **commits** tworzone są pliki o nazwach repozytoriów, do których użytkownicy wykonują commity. Każdy rekord zapisywany w pliku danego repozytorium oznaczany jest aktualnym znacznikiem czasu oraz adresem IP, z którego commit został przesłany.

Katalog **repos** zawiera repozytoria projektów będące wynikiem operacji **push** wykonanej przez klienta. Dane w tym katalogu są każdorazowo aktualizowane podczas przesyłania nowej wersji repozytorium.

Ponadto serwer tworzy własny plik o nazwie **list**, który zawiera listę wszystkich repozytoriów obsługiwanych przez system. Plik ten pełni funkcję rejestru dostępnych projektów.

Dodatkowo w systemie istnieje katalog **backups**, w którym przechowywane są kopie zapasowe repozytoriów. W momencie nadpisania repozytorium o podanej przez użytkownika nazwie jego poprzednia wersja zapisywana jest w tym katalogu.

Istotnym elementem projektu jest rola administratora systemu (operatora serwera), który odpowiada za zarządzanie zasobami serwera. W przypadku podjęcia decyzji o zakończeniu rozwoju danego projektu administrator musi ręcznie usunąć wpis repozytorium z pliku **list**, katalog repozytorium z **repos** oraz plik z **commits**, a w razie potrzeby, odpowiadającą mu kopię zapasową z katalogu **backups**.

Aplikacja serwera

Administrator zarządza przebiegiem realizacji projektów, nadzorując pracę programistów za pośrednictwem logów systemowych (syslog). Odpowiada on również za zarządzanie strukturą danych serwera, w tym usuwanie wybranych katalogów lub pojedynczych wpisów po wcześniejszej konsultacji z zespołem projektowym.

Pod bezpośrednim nadzorem administratora znajdują się następujące elementy struktury danych serwera:

- Katalog **/TIG/srv/data/repos** zawiera najnowsze wersje repozytoriów, pochodzące z ostatniej operacji **push** wykonanej przez użytkowników.
- Katalog **/TIG/srv/data/commits** stanowi zbiór plików nazwanych zgodnie z repozytoriami, w których zapisywane są informacje o commitach, obejmujące znacznik czasu oraz adres IP użytkownika, który wykonał operację commit.
- Plik **/TIG/srv/data/list** zawiera listę repozytoriów przesłanych do systemu i pełni funkcję centralnego rejestru projektów.
- Katalog **/TIG/srv/data/backups** przechowuje kopie zapasowe repozytoriów, tworzone na podstawie poprzednich wersji danych przed wykonaniem kolejnego **push**.

Aplikacja klienta

Aplikacja kliencka umożliwia użytkownikowi cztery podstawowe sposoby interakcji z serwerem TIG, realizowane za pomocą dedykowanych komend wywoływanych z poziomu wiersza poleceń.

- Komenda **pull** pozwala na pobranie repozytorium o nazwie wskazanej w drugim argumencie i zapisanie go w bieżącym katalogu roboczym systemu (cwd).
- Komenda **push** służy do wysłania do zasobów serwera katalogu projektu o podanej nazwie, znajdującego się w bieżącej lokalizacji roboczej użytkownika.
- Komenda **discover** umożliwia odnalezienie serwera TIG w sieci lokalnej z wykorzystaniem protokołu UDP w trybie multicast.
- Komenda **commit** pozwala na dodanie komentarza do wskazanego repozytorium. Użytkownik przekazuje nazwę repozytorium oraz treść komentarza, który zostaje zapisany na serwerze jako nowy wpis w historii commitów.
- Komenda **repos**, wysyła zapytanie do serwera w celu uzyskania listy wszystkich dostępnych repozytoriów.
- Komenda **read**, wysyła zapytanie do serwera w celu uzyskania rekordów w /TIG/srv/data/commits dla repozytorium o wskazanej nazwie (drugi argument).

Komenda **commit** została celowo oddzielona od komendy **push**. Umożliwia to dodawanie wielu komentarzy do jednej wersji projektu, co pozwala na komunikację w zespole bez konieczności wysyłania na serwer „pustych" aktualizacji repozytoriów, które nie zawierają zmian w kodzie. Umożliwia to także stworzenie "wirtualnego repozytorium" - takiego bez katalogu projektu w **/TIG/srv/data/repos** oraz kopii zapasowej w **/TIG/srv/data/backups**, z samymi rekordami w **/TIG/srv/data/commits**. Wirtualne repozytorium pełni rolę komunikatora wewnątrz zespołu. Przykład użycia poniżej.

```
root@0a0ec4522820:/tmp# ./TIG_cli commit chat wiadomosc do czlonkow zespolu
OK
root@0a0ec4522820:/tmp# ./TIG_cli read chat
2026-01-02 13:33:11: HOST: fc00:2::2, MESSAGE: wiadomosc do czlonkow zespolu
OK
root@0a0ec4522820:/tmp#
```

Przykład działania

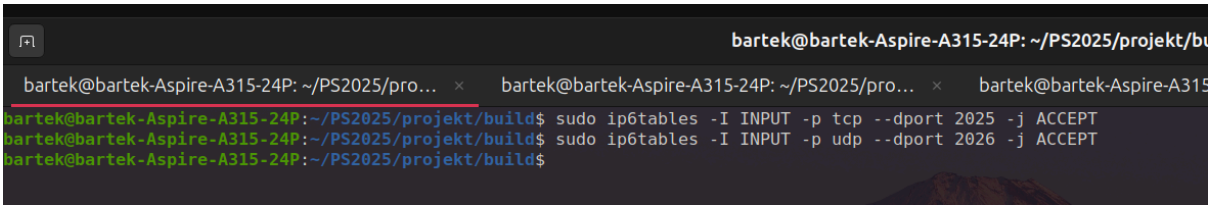
Administrator:

1. Sprawdza na którym interfejsie obsługującym multicast znajduje się adres IPv6 link-global (potrzebny do połączenia z klientem). Można w razie potrzeby dodać stosowny adres IPv6.

```
bartek@bartek-Aspire-A315-24P:~/PS2025/projekt/build$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: wlp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 9c:2f:9d:83:88:2f brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.242/24 brd 192.168.100.255 scope global dynamic noprefixroute wlp2s0
       valid_lft 83939sec preferred_lft 83939sec
    inet6 fe80::cc2:78b2:c8b5:597e/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: br-50ca92fdda97: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 2a:aa:77:64:31:bd brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.1/24 brd 192.168.1.255 scope global br-50ca92fdda97
       valid_lft forever preferred_lft forever
    inet6 fc00:2::1/64 scope global nodad
       valid_lft forever preferred_lft forever
    inet6 fe80::28aa:77ff:fe64:31bd/64 scope link
       valid_lft forever preferred_lft forever
4: br-9a156c9b338e: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether b6:da:1b:54:66:35 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.1/24 brd 192.168.56.255 scope global br-9a156c9b338e
       valid_lft forever preferred_lft forever
    inet6 fc00:1::1/64 scope global nodad
       valid_lft forever preferred_lft forever
    inet6 fe80::b4da:1bff:fe54:6635/64 scope link
       valid_lft forever preferred_lft forever
```

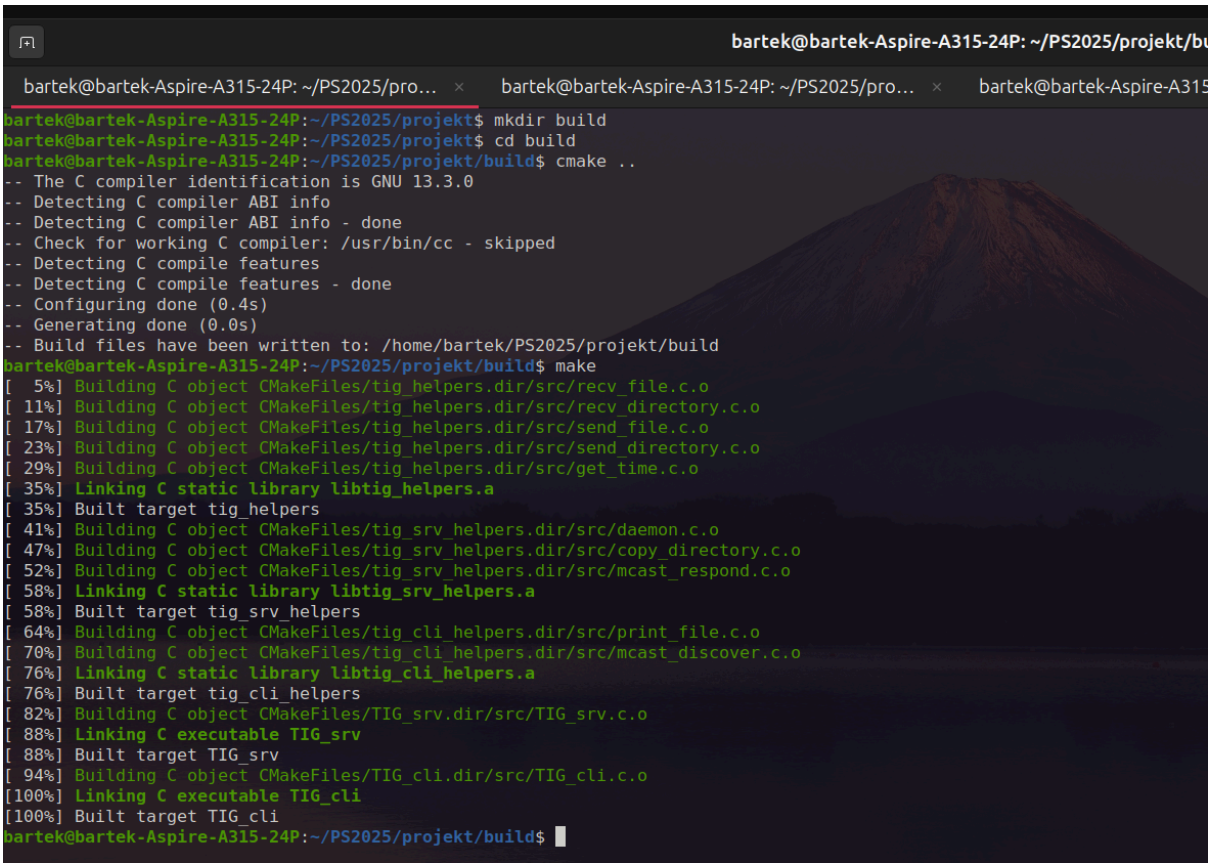
W prezentacji został użyty interfejs **3: br-50ca92fdda97**, interfejs można zmienić (makro **MCAST_IF** w pliku **mcast_respond.h**).

2. Wyłącza firewall na portach używanych przez system TIG.

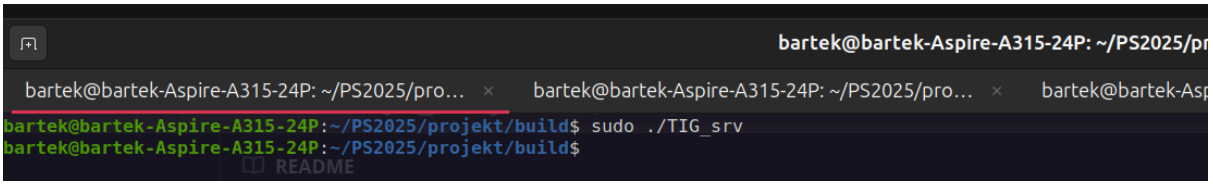


TCP port 2025 - połączenie z klientem, wysyłanie katalogów.
UDP port 2026 - mechanizm multicast discovery.

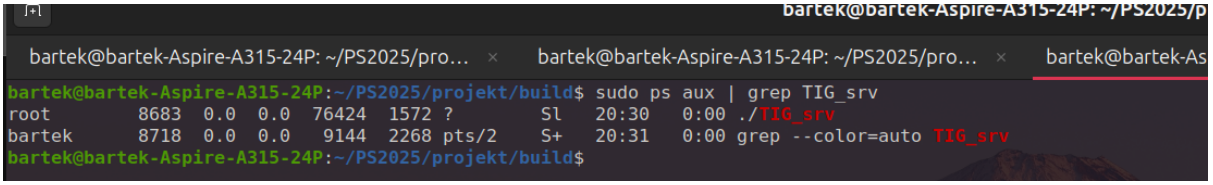
3. Buduje projekt za pomocą CMake.



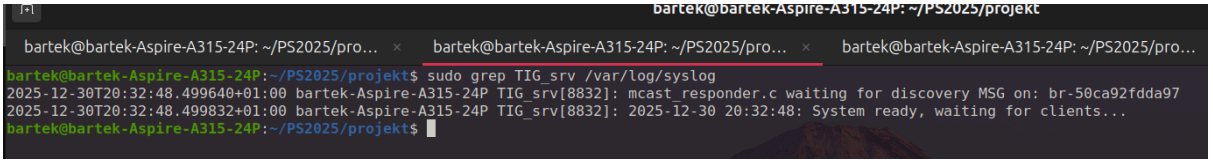
4. Uruchamia proces serwera przy użyciu sudo.



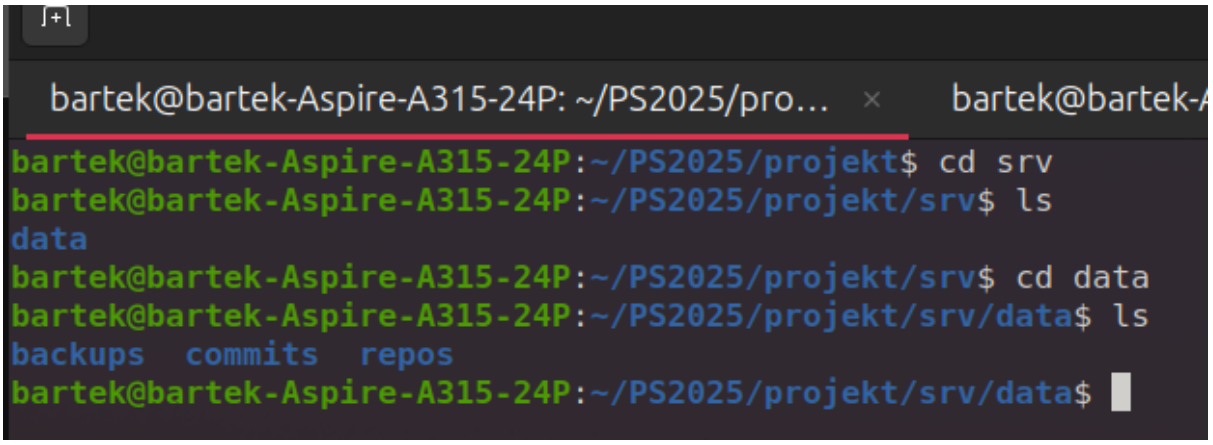
5. Można sprawdzić czy proces demona wystartował:



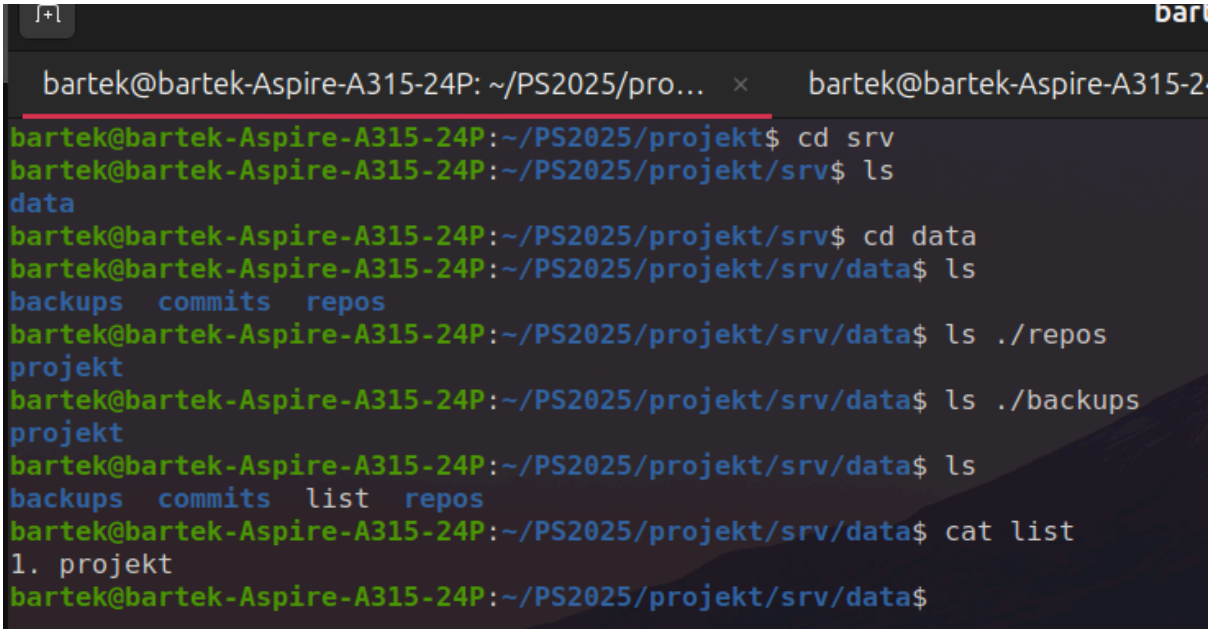
6. I obejrzeć pierwsze logi serwera:



7. Struktura danych serwera omawiana w poprzednich częściach dokumentacji wygląda następująco:



8. Po pierwszym push od klienta:



9. Przykładowy rekord w commits:

```
bartek@bartek-Aspire-A315-24P:~/PS2025/projekt/srv/data$ cd commits/
bartek@bartek-Aspire-A315-24P:~/PS2025/projekt/srv/data/commits$ ls
projekt
bartek@bartek-Aspire-A315-24P:~/PS2025/projekt/srv/data/commits$ cat projekt
2025-12-30 20:41:58: HOST: fc00:2::2, MESSAGE: hello
bartek@bartek-Aspire-A315-24P:~/PS2025/projekt/srv/data/commits$
```

10. Logi w syslog:

```
bartek@bartek-Aspire-A315-24P:~/PS2025/projekt$ sudo grep TIG_srv /var/log/syslog
[sudo] password for bartek:
2025-12-30T20:32:48.499640+01:00 bartek-Aspire-A315-24P TIG_srv[8832]: mcast_responder.c waiting for discovery MSG on: br-50ca92fdda97
2025-12-30T20:32:48.499832+01:00 bartek-Aspire-A315-24P TIG_srv[8832]: 2025-12-30 20:32:48: System ready, waiting for clients...
2025-12-30T20:39:05.456395+01:00 bartek-Aspire-A315-24P TIG_srv[8832]: mcast_respond.c recieved discovery MSG: TIG_DISCOVERY
2025-12-30T20:39:05.457495+01:00 bartek-Aspire-A315-24P TIG_srv[8832]: mcast_respond.c sent discovery response.
2025-12-30T20:39:53.474546+01:00 bartek-Aspire-A315-24P TIG_srv[9013]: 2025-12-30 20:39:53: HOST: fc00:2::2, pushed: projekt
2025-12-30T20:41:58.541472+01:00 bartek-Aspire-A315-24P TIG_srv[9078]: 2025-12-30 20:41:58: HOST: fc00:2::2, commited: projekt
bartek@bartek-Aspire-A315-24P:~/PS2025/projekt$
```

Aby zamknąć proces serwera, należy sprawdzić jego PID i wykonać komendę **sudo kill -9 <PID serwera TIG>.**

Klient:

1. Sprawdza który interfejs obsługuje multicast i oferuje adres IPv6 link-local i link-global (jeśli takiego adresu nie ma, to można go dodać).

```
root@0a0ec4522820:/tmp# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.2  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::4ca5:d3ff:fe8a:e409  prefixlen 64  scopeid 0x20<link>
        inet6 fc00:1::2  prefixlen 64  scopeid 0x0<global>
        ether 4e:a5:d3:8a:e4:09  txqueuelen 0  (Ethernet)
        RX packets 31  bytes 4355 (4.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 20  bytes 1540 (1.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.2  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fc00:2::2  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::fca0:abff:fec1:23e5  prefixlen 64  scopeid 0x20<link>
        ether fe:a0:ab:c1:23:e5  txqueuelen 0  (Ethernet)
        RX packets 74  bytes 8273 (8.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 57  bytes 4839 (4.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

W prezentacji został użyty interfejs: **eth1**, interfejs można zmienić (makro **MCAST_IF** w pliku **mcast_discover.h**).

2. Znajduje adres IPv6 do komunikacji z serwerem TIG.

```
root@0a0ec4522820:/tmp# ./TIG_cli discover
Interface: br-50ca92fdda97 IPv6 server addresses:
fc00:2::1
fe80::28aa:77ff:fe64:31bd
root@0a0ec4522820:/tmp#
```

3. Dopisuje do pliku **/etc/hosts** rekord DNS

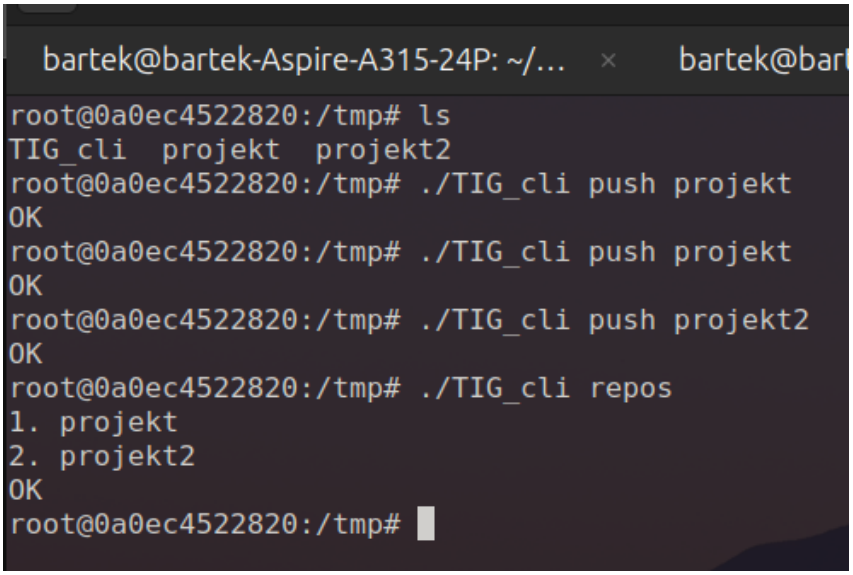
```
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::  ip6-localnet
ff00::  ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.168.56.2    0a0ec4522820
fc00:1::2       0a0ec4522820
192.168.1.2     0a0ec4522820
fc00:2::2       0a0ec4522820
172.18.0.2      0a0ec4522820
fc00:2::1       TIG_srv
```

Użytkownik może zmienić nazwę domeny, ale wymaga to zmiany makra SERVER_NAME w pliku /include/TIG_cli.h

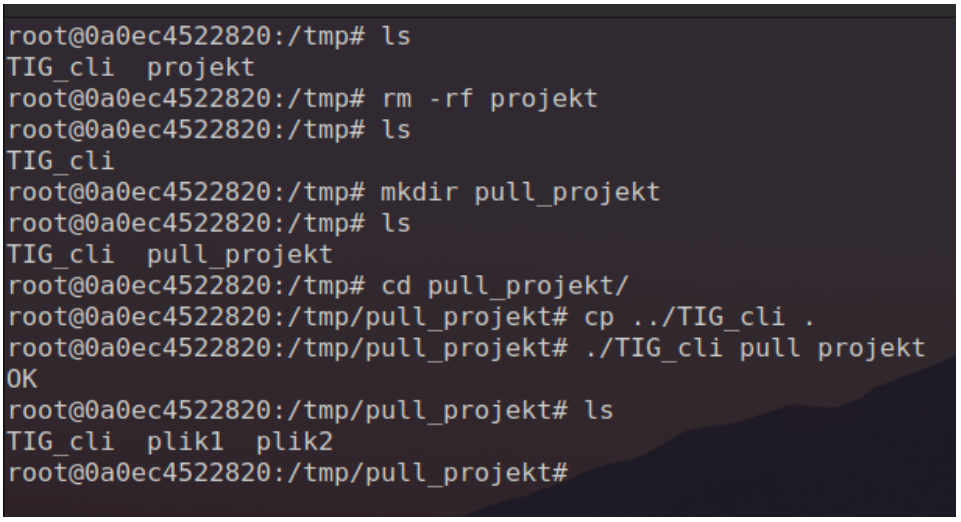
4. Pushuje i commituje:

```
root@0a0ec4522820:/tmp# ./TIG_cli push projekt
OK
root@0a0ec4522820:/tmp# ./TIG_cli commit projekt hello
OK
root@0a0ec4522820:/tmp#
```

5. Sprawdza dostępne repozytoria:



6. Pulluje dostępne repozytorium (w pierwszej kolejności należy utworzyć katalog, do którego będziemy pobierać pliki i katalogi):



W terminalu klienta można również ustawić **alias**, tak aby korzystać z aplikacji klienta z dowolnego miejsca w systemie:

```
root@0a0ec4522820:/tmp# vim ~/.bashrc
root@0a0ec4522820:/tmp#
```

```
# ~/.bashrc: executed by bash(1) for non-login shells.

# Note: PS1 is set in /etc/profile, and the default umask is defined
# in /etc/login.defs. You should not need this unless you want different
# defaults for root.
# PS1='${debian_chroot:+($debian_chroot)}\h:\w\$ '
# umask 022

# You may uncomment the following lines if you want `ls' to be colorized:
# export LS_OPTIONS='--color=auto'
# eval "$(dircolors)"
# alias ls='ls $LS_OPTIONS'
# alias ll='ls $LS_OPTIONS -l'
# alias l='ls $LS_OPTIONS -lA'
#
# Some more alias to avoid making mistakes:
# alias rm='rm -i'
# alias cp='cp -i'
# alias mv='mv -i'
alias TIG='/tmp/TIG_cli'
                                                    20,24         All
```

Projekt, instrukcja obsługi, oraz dokumentacja wszystkich funkcji, w formie komentarzy, znajduje się na: https://github.com/BarOs0/TIG