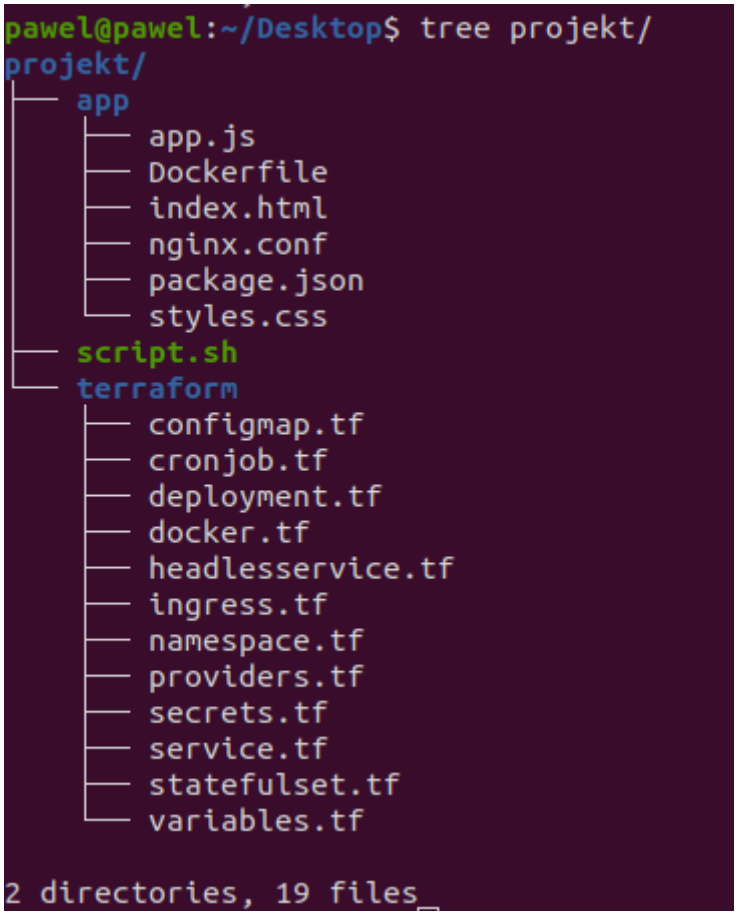
Projekt indywidualny

WYKONANY NA OCENĘ 4.0
POPRAWIONA WERSJA

PAWEŁ MUSZYŃSKI 48770

Projekt zrealizowany w ramach oceny na 4.0 obejmuje aplikację podzieloną na dwa główne podfoldery:

- **app** – zawiera wszystkie kluczowe pliki aplikacji, a także konfigurację serwera **nginx**, która zapewnia poprawne działanie aplikacji w środowisku produkcyjnym.
- **terraform** – zawiera pliki odpowiedzialne za konfigurację infrastruktury



Problem z minikube w terraformie oraz sprytne rozwiązanie go:

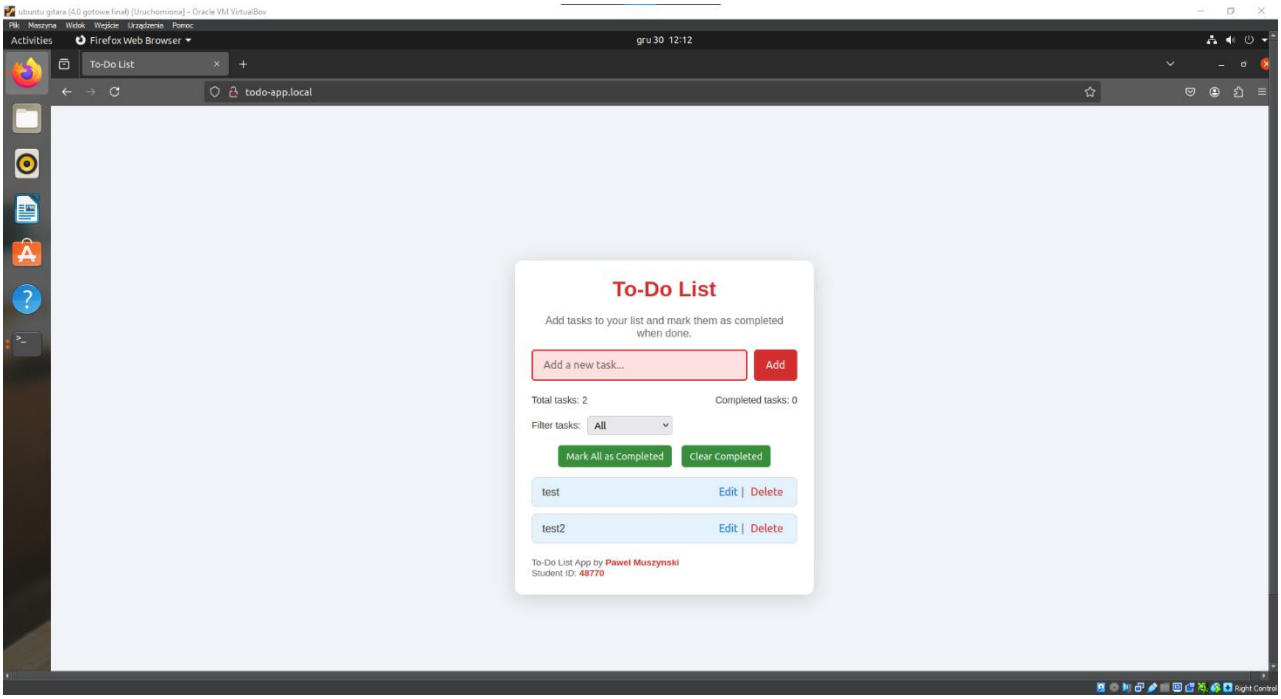
W moim projekcie napotkałem problem z Minikube w Terraformie. Okazało się, że pliki załączone w Moodle w jakiś sposób nie działały poprawnie. Gdy próbowałem uruchomić polecenia terraform init i terraform apply, występowały błędy typu „no route to host”. Jednak kiedy uruchamiałem te same polecenia ręcznie, poza Terraformem, bez użycia null_resource, działały one bez problemu i strona internetowa była wyświetlana prawidłowo.

1. **Script.sh**

Skrypt uruchamia Minikube z minikube start --driver=docker, włączając dodatek Ingress do zarządzania ruchem HTTP. Następnie przechodzi do katalogu aplikacji, ustawia kontekst kubectl na Minikube oraz konfiguruje środowisko Docker w Minikube, aby budować obrazy. Po zakończeniu tego etapu wraca do głównego katalogu, przechodzi do katalogu Terraform, inicjalizuje go i aplikuje zmiany w konfiguracji infrastruktury.

Krótkie omówienie plików z folderu app:

1. **index.html**
    - Zawiera główny kod aplikacji **To-Do List**, w tym strukturę interfejsu użytkownika.



2. **styles.css**
    - Plik odpowiedzialny za stylizację aplikacji.
3. **app.js**
    - Kluczowy plik logiki aplikacji To-Do. Funkcje, które zawiera:
        - **Obsługa LocalStorage**: Przechowywanie danych zadań w przeglądarce, co pozwala zachować listę między sesjami.
        - **Dynamiczne renderowanie zadań**: Lista zadań jest generowana na podstawie danych w LocalStorage, z możliwością filtrowania zadań ukończonych i nieukończonych.
        - **Dodawanie nowych zadań**: Użytkownik może dodać nowe zadanie poprzez wprowadzenie tekstu i kliknięcie przycisku.
        - **Edycja i usuwanie zadań**: Opcje edycji pozwalają zmienić treść zadania, a usunięcie pozwala je usunąć z listy.
        - **Licznik zadań**: Wyświetla liczbę wszystkich zadań oraz tych ukończonych.
        - **Czyszczenie ukończonych zadań**: Przycisk usuwa wszystkie zadania oznaczone jako ukończone.
4. **Dockerfile**

- Składa się z dwóch etapów:
    - **Etap budowania aplikacji**: Wykorzystuje obraz Node.js do instalacji zależności i skopiowania kodu aplikacji.
    - **Etap produkcyjny**: Używa obrazu NGINX do serwowania aplikacji.
    - Zmienna środowiskowa jest dynamicznie wstawiana do pliku index.html przed uruchomieniem serwera.

5. **nginx.conf**
    - Plik konfiguracji serwera NGINX, który obsługuje:
        - Pliki statyczne (HTML, CSS, JS).
        - Buforowanie plików, aby zwiększyć wydajność.
        - Fallback dla ścieżki /, zapewniając, że aplikacja działa poprawnie jako aplikacja jednopodstronowa (SPA).
6. **package.json**
    - Plik definiujący metadane aplikacji oraz jej zależności.
    - Minimalistyczna konfiguracja zawiera nazwę, wersję oraz skrypty, takie jak npm test.

Krótkie omówienie plików z folderu app:

Konfiguracja zawiera kompletny zestaw zasobów Kubernetes zdefiniowanych w plikach Terraform. Wdrożenie infrastruktury przebiega bezproblemowo, a aplikacja uruchamia się poprawnie po wykonaniu komendy **terraform apply**.

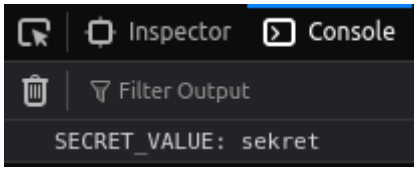
```
Apply complete! Resources: 13 added, 0 changed, 0 destroyed.
```

Kluczowe zasoby:

1. **Ingress:**
    - Udostępnia aplikację webową todo-app w namespace namespace-number1 pod hostem todo-app.local.
    - Obsługuje ścieżkę / i kieruje ruch do usługi app-service.
2. **Service myapp-db-service:**
    - Headless Service dla StatefulSet PostgreSQL, obsługujący ruch na porcie 5432.
3. **StatefulSet myapp-db:**
    - Tworzy bazę danych PostgreSQL w namespace namespace-number1.
    - Korzysta z PersistentVolumeClaim dla przechowywania danych.
4. **Deployment app:**

- Wdraża aplikację todo-app z dwoma replikami w namespace namespace-number1.
- Pobiera konfigurację z ConfigMap oraz zmienne środowiskowe z Secret.
- Limity zasobów kontenerów zostały jasno określone.

5. **CronJob:**
   - W namespace namespace-number2, uruchamiany co minutę, wykonuje proste zadanie logowania daty.
6. **Namespaces:**
   - Izolacja zasobów została zapewniona poprzez utworzenie dwóch namespaces: namespace-number1 i namespace-number2.
7. **ConfigMap:**
   - Przechowuje kluczowe ustawienia aplikacji, takie jak:
     - Nazwa aplikacji,
     - Tryb działania,
     - Ustawienia sieciowe.
8. **Secrets:**
   - Tworzy Secret w namespace namespace-number1, przechowujący klucz SECRET_VALUE o wartości zakodowanej w base64.



9. **Providers:**
   - Definiuje provider Kubernetes, który łączy się z klastrem Minikube za pomocą certyfikatów i odpowiednich ustawień konfiguracji.
10. **Variables:**

- Zmienna service_ports definiuje listę portów, które są dynamicznie wykorzystywane w konfiguracji usług w Kubernetes (np. porty 80, 443, 5432).

11. **Docker:**

- Łączy się z lokalnym demonem Dockera
- Buduje obraz my_app:latest z katalogu ../app zawierającego Dockerfile.
- Ładuje obraz do Minikube za pomocą komendy minikube image load.
- Tworzy kontener my_app_container z obrazem my_app:latest, wystawia port 8080 i ustawia zmienną środowiskową SECRET_VALUE

Wdrożenie aplikacji:

- **Aplikacja todo-app** została wdrożona jako **Deployment** z określonymi zasobami (CPU, memory) i dostępem do konfiguracji z **ConfigMap**.
- **StatefulSet** z bazą **PostgreSQL** wykorzystuje **PersistentVolumeClaim** do przechowywania danych.
- Ruch do aplikacji webowej jest przekierowywany za pomocą **Ingress**.
- **CronJob** w drugim namespace demonstruje harmonogramowanie zadań w Kubernetes.
- Dodanie linijki *192.168.49.2 todo-app.local* w pliku hosts, który znajduje się w katalogu /etc. Dzięki temu możliwe jest wchodzenie na stronę przez adres **todo-app.local** zamiast korzystania z numeru IP.

Aby prawidłowo uruchomić aplikację należy uruchomić skrypt, który tak jak wyżej jest opisane na początku uruchomi minikube ustawiając najważniejsze zależności, a następnie wykona terraform init oraz apply

Logi potwierdzające działanie plików terraform (prawidłowe skonfigurowanie):

Zasoby:

```
pawel@pawel:~/Desktop/projekt$ kubectl get all -n namespace-number1
NAME                                 READY   STATUS    RESTARTS      AGE
pod/app-deployment-698b8468cf-ggwrn  1/1     Running   2 (22m ago)   14d
pod/app-deployment-698b8468cf-phmht  1/1     Running   2 (22m ago)   14d
pod/myapp-db-0                       1/1     Running   2 (22m ago)   14d

NAME                     TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                                  AGE
service/app-service      NodePort    10.104.234.114   <none>        80:30429/TCP,443:31633/TCP,5432:32318/TCP   14d
service/myapp-db-service ClusterIP   None             <none>        5432/TCP                                 14d

NAME                             READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/app-deployment   2/2     2            2           14d

NAME                                        DESIRED   CURRENT   READY   AGE
replicaset.apps/app-deployment-698b8468cf   2         2         2       14d

NAME                     READY   AGE
statefulset.apps/myapp-db   1/1     14d
```

```
pawel@pawel:~/Desktop/projekt$ kubectl get all -n namespace-number2
NAME                                  READY   STATUS              RESTARTS   AGE
pod/simple-cronjob-28925971-rwj24     0/1     Completed           0          3m2s
pod/simple-cronjob-28925972-59m59     0/1     Completed           0          2m2s
pod/simple-cronjob-28925973-bmhsr     0/1     Completed           0          62s
pod/simple-cronjob-28925974-7ndc6     0/1     ContainerCreating   0          2s

NAME                           SCHEDULE    TIMEZONE   SUSPEND   ACTIVE   LAST SCHEDULE   AGE
cronjob.batch/simple-cronjob   * * * * *   <none>     False     1        2s              14d

NAME                                  STATUS     COMPLETIONS   DURATION   AGE
job.batch/simple-cronjob-28925971     Complete   1/1           6s         3m2s
job.batch/simple-cronjob-28925972     Complete   1/1           7s         2m2s
job.batch/simple-cronjob-28925973     Complete   1/1           5s         62s
job.batch/simple-cronjob-28925974     Running    0/1           2s         2s
```

Deployment:

```
pawel@pawel:~/Desktop/projekt$ kubectl get deployments -n namespace-number1
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
app-deployment   2/2     2            2           14d
```

Configmap:

```
pawel@pawel:~$ kubectl describe configmap app-config -n namespace-number1
Name:         app-config
Namespace:    namespace-number1
Labels:       <none>
Annotations:  <none>

Data
====
CACHE_TIMEOUT:
----
300

CONTAINER_PORT:
----
80

DB_PORT:
----
5432

LOG_LEVEL:
```

Obraz, który jest wrzucany do minikube w pliku docker.tf :

```
pawel@pawel:~$ minikube ssh
docker@minikube:~$ docker images
REPOSITORY                                       TAG       IMAGE ID       C
REATED         SIZE
todo-app                                         latest    6e342e4eb6f2   4
 hours ago     192MB
my_app                                           latest    a06c8505c336   5
 hours ago     192MB
postgres                                         14        15ca0e19f324   8
 weeks ago     422MB
busybox                                          latest    af4709625109   3
 months ago    4.27MB
registry.k8s.io/ingress-nginx/controller         <none>    a80c8fd6e522   5
 months ago    287MB
registry.k8s.io/ingress-nginx/kube-webhook-certgen   <none>    ce263a8653f9   5
```