Projektowanie i wdrażanie systemów w chmurze

Lista zadań na pracownię 24.11.17

Aby wygenerować klucze dostępu do GCP, które będą potrzebne Terraformowi do dostępu do API, z menu konsolki GCP wybierzcie “APIs & services -> Credentials”. Tam użyjcie “Create credentials -> Service account key”. Następnie wybierzcie “New service account” i nadajcie mu nazwę. Aby mieć gwarancję, że klucze pozwolą na zarządzanie wszelkimi zasobami w projekcie, w polu “Role” można użyć “Project -> Editor”. Rodzaj klucza, jaki potrzebujemy, to “JSON”. Po utworzeniu zapiszcie przygotowany przez Google plik .json w wybranym miejscu, i ustawcie zmienną środowiskową GOOGLE_APPLICATION_CREDENTIALS na ścieżkę do tego pliku - terraform będzie używać tej zmiennej, aby dosięgnąć kluczy do API.

Aby rozpocząć nowy “projekt” z Terraformem wystarczy utworzyć pusty katalog i umieścić w nim plik(i) .tf. Komenda terraform init jest konieczna przed innymi poleceniami (acz terraform sam da znać, kiedy potrzebny jest mu init, gdybyśmy zapomnieli lub dodali do projektu nowych providerów). Pamiętajcie, że na wykładzie pokazywałem Terraform w wersji 0.11.0.

Pracując z Terraformem będziemy używać głównie dwóch komend: terraform plan, która wyświetla jakie zmiany Terraform chciałby wykonać w tym momencie, bez wykonywania ich, oraz terraform apply, która te zmiany faktycznie wprowadza. Plan generowany komendą *plan* można zapisać do pliku używając flagi *-out=plik-z-planem*. Aby użyć taki zapisany plan w komendzie *apply,* wystarczy go podać jako kolejny argument, tj. *terraform apply plik-z-planem*.

Aby skasować całą infrastrukturę do zera (można usunąć wszystkie pliki .tf, a wtedy plan/apply zaproponują skasowanie wszystkiego do zera, zgodnie z aktualnym stanem pustej konfiguracji - ale fajniej jest nie usuwać całej konfiguracji) wygodna jest flaga *-destroy* do komendy *plan*, która wygeneruje plan usuwający wszystkie zasoby. Taki plan trzeba zapisać do pliku (komenda *apply* nie zna flagi *destroy*). To wygodny sposób aby “posprzątać po zajęciach” i oszczędzać creditsy. Oczywiście, po wykonaniu takiego planu następny *plan* będzie proponował postawienie wszystkiego od nowa, co pozwala wygodnie wrócić do kompletnego stanu.

Tym razem zadania są naprawdę proste. Tekstu jest dużo, by sprecyzować wymagania.

1. [6 pkt] Chcielibyśmy stworzyć prostą infrastrukturę składającą się z serwera www i chmurowej bazy danych. Opisz przy pomocy Terraforma (zalecamy wykorzystanie modułów, ale w tym zadaniu nie jest to konieczne) konfigurację zasobów chmurowych mając na uwadze następujące wymagania:
    a. Serwer powinien posiadać prywatny i publiczny adres IP. Prywatny adres IP powinien należeć do specjalnie stworzonej podsieci dla serwerów www.
    b. Firewall w GCP powinien być ustawiony tak, aby serwer www akceptował wyłącznie połączenia:
        i. Na porcie 80 i 443 z całego Internetu
        ii. Na porcie 22 z określonej grupy adresów IP (np. Tylko z Twojego adresu IP i sieci w Instytucie Informatyki UWr: 156.17.4.0/24). Miej na uwadze to, że możesz stracić dostęp jeśli zmieni się Twój adres IP.
    c. Jako baza danych powinna zostać wykorzystana usługa Cloud SQL. Powinna ona akceptować wyłącznie połączenia przychodzące z publicznego adresu IP serwera www (wykorzystaj fakt, że tworząc serwer www poznasz jego adres IP i możesz przekazać go jako parametr do konfiguracji firewalla dla bazy danych).

2. [6 pkt] Opiszmy Terraformem konfigurację dobrze znanego nam z wcześniejszych pracowni środowiska: load-balancer HTTP + kilka serwerów aplikacji. W tym celu przygotujmy dwa moduły, które mogą się przydać w przyszłości: jeden moduł będzie opisywał konfigurację load-balancera, drugi będzie przygotowywał serwery aplikacji. Specyfikacja argumentów (variables) i atrybutów (output) modułów powinna być taka, aby dało się je wygodnie wykorzystać w różnych scenariuszach, ale w szczególności:
    a. Moduł load-balancera powinien:
        - Przyjmować argument podający listę adresów IP serwerów, do których load-balancer ma kierować zapytania.
        - Automatycznie instalować load-balancer na serwerze oraz konfigurować go według podanej listy adresów IP.
        - Zarezerwować publiczny adres IP i przygotować podsieć, w której będzie umieszczona instancja serwera oraz właściwe dla load-balancera reguły zapory ogniowej.

        - Adres IP, na którym dostępny jest load-balancer, udostępnić jako wyjście modułu.
    b. Moduł serwerów aplikacji powinien:
        - Przyjmować argumenty określające: pożądaną liczbę serwerów, rodzaj instancji na serwery, ścieżkę do katalogu (na lokalnej maszynie, tj. tej, na której uruchamiany jest Terraform) w której znajdują się pliki strony, która powinna być zamieszczona na serwerach.
        - Automatycznie instalować serwer HTTP na tworzonych serwerach oraz zamieszczać na nich pliki strony, którą serwer będzie udostępniał.
        - Przygotować podsieć oraz rozsądne reguły zapory ogniowej.
        - Lista adresów IP wszystkich serwerów aplikacji musi być udostępniona jako wyjście z tego modułu.

    Zastanów się, czy nie ma więcej argumentów/wyników, które warto, by te moduły obsługiwały.
    W obu przypadkach do przygotowania serwerów nie trzeba (ale można, patrz zad. 3) używać Ansible - zazwyczaj wystarczy krótki skrypt. Aby wypróbować oba moduły, przygotuj prostą konfigurację, która używa obu z nich, podając im przykładowe wartości argumentów oraz łącząc je w jeden współpracujący system.
3. * [1 pkt extra] W zadaniu do provisioningu 2 wykorzystaj konfigurację Ansible, najlepiej z poprzedniej pracowni.