Projektowanie i wdrażanie systemów w chmurze

Lista zadań na pracownię 2018.11.12

Aby wygenerować klucze dostępu do chmury, które będą potrzebne Terraformowi do dostępu do API, należy:

- **AWS** - po prostu ustawić Wasz AWS_ACCESS_KEY_ID oraz AWS_SECRET_ACCESS_KEY w ~/.aws/credentials oraz ustawić zmienną środowiskową AWS_PROFILE - tak samo jak na pierwszej pracowni.
- **GCP** - 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). Pracując z Terraformem będziemy używać głównie komendy: terraform apply, która wprowadza w życie zmiany opisane w plikach .tf. Na wykładzie pokazywałem Terraform w wersji 0.11.10.

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 komenda terraform destroy, która wygeneruje plan odwrotny, usuwający wszystkie zasoby. 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 kompletnej konfiguracji.

1. * [1 pkt extra] Poniższe zadania zrób w różnych chmurach.

2. [6 pkt] Chcielibyśmy stworzyć prostą infrastrukturę składającą się z serwera www i chmurowej bazy danych. Opisz przy pomocy Terraforma konfigurację zasobów chmurowych.
   a. Serwer powinien posiadać prywatny i publiczny adres IP. Prywatny adres IP powinien należeć do specjalnie stworzonej podsieci dla serwerów www.
   b. Security group / firewall powinien być ustawiony tak, aby serwer www akceptował wyłącznie połączenia:
      - na porcie 80 i 443 z całego Internetu oraz
      - 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).
   c. Jako bazę danych należy użyć usługi RDS lub Cloud SQL. Baza danych musi 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 security group / firewalla bazy danych).

3. [6 pkt] Opisz Terraformem konfigurację środowiska z poprzedniej pracowni: load-balancer HTTP + kilka serwerów aplikacji. W tym celu przygotuj 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 skonfigurować go według podanej listy adresów IP.
      - Zarezerwować publiczny adres IP i przygotować podsieć, w której będzie umieszczona instancja serwera oraz właściwe dla takiego 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 zainstalować 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!) używać Ansible[1].
Aby wypróbować oba moduły, napisz prostą konfigurację chmury, która używa obu modułów, podając im przykładowe wartości argumentów oraz łącząc je w jeden współpracujący system.

4. * [2 pkt extra] W zadaniu 3 zamiast konfigurować serwery po ich uruchomieniu, uruchamiaj je od razu z gotowego, uprzednio przygotowanego obrazu / AMI (nieważne jak go utworzysz). Zauważ, że listy adresów serwerów www nie możesz wypiec na stałe w obrazie load-balancera - ten parameter zmienia się w zależności od konfiguracji infrastruktury. Użyj parametru *user-data* instancji (lub podobnego mechanizmu), by przekazać listę IP serwerów www[2].

[1] Nie będzie to oceniane, ale może ułatwić pracę.
[2] Jeżeli uznasz to za konieczne, możesz użyć własnego skryptu, który zinterpretuje dane w *user-data*.