Projektowanie i wdrażanie systemów w chmurze

Lista zadań na pracownię 27.10.17

W zadaniach na tej liście uruchamiamy strony internetowe przez HTTP, ale kompletnie nie interesuje nas treść na tych stronach - mogą być brzydkie i niedopracowane. Skupiamy się na uruchamianiu usług chmurowych, komunikacji z serwerami i konfigurowaniu ich. Jednym z celów tej listy zadań jest by każdy student samemu odkrył skąd mogą brać się niewygody ręcznego zarządzania serwerami. W poniższych zadaniach mówiąc “serwer” mamy oczywiście na myśli *VM instance* w *Google Compute Engine*.

1. [2 pkt.] Zainstaluj serwer HTTP (np. *apache2* lub *nginx*) na serwerze i zamieść na nim prostą, ale nie całkiem statyczną, stronę, złożoną np. z prostego pliku *.php* oraz 2 obrazków i jakiegoś szablonu *.css*. Jeżeli nie lubisz PHP, możesz wybrać dowolny inny język, byleby było widać, że treść strony nie jest statyczna (niech np. wyświetla losowy tekst lub licznik odwiedzin - ale nie skupiamy się na programowaniu ciekawych aplikacji). Obrazki oraz CSS trzymaj nie na serwerze, tylko np. w GoogleCloudStorage i wstaw odpowiednie łącza w dokument. Upewnij się, że strona działa poprawnie i jest publicznie dostępna.
   Zastanów się, jakie narzędzia mogłyby ułatwić Ci publikację strony na serwer / CloudStorage.
2. [2 pkt.] Uruchom nieduży serwer i zainstaluj na nim porządnego CMSa np. *Wordpress* lub *Drupal*. Możesz też wybrać inny, ale niech przynajmniej używa bazy danych. Nie używaj gotowych obrazów serwerów z zainstalowanym CMSem, użyj serwera z “czystym” distro linuksowym i samemu zainstaluj i skonfiguruj CMS. Do bazy danych użyj właściwej usługi chmurowej, nie instaluj jej samemu na serwerze. Upewnij się, że całość działa poprawnie i da się sterować CMSem z poziomu strony internetowej, ale nie musisz do niego wprowadzać żadnej konkretnej treści.
3. [3 pkt.] Na dwóch-trzech serwerach uruchom identyczną, prostą aplikację/stronę HTTP (może np. być taka sama, jak w zadaniu 1.). Dodaj jeszcze jeden serwer, na którym uruchomisz własny load-balancer HTTP (*haproxy* lub *nginx* w trybie *reverse-proxy*), który otrzymane zapytania będzie, w miarę równomiernie, przekierowywał do jednego z właściwych serwerów. Zauważ, że serwery z aplikacją nie muszą być publicznie dostępne, wystarczy, że load-balancer będzie. Upewnij się, że strona jest dostępna przez load-balancer i działa poprawnie, oraz że wszystkie serwery aplikacji obsługują zapytania. Następnie opublikuj “nową wersję aplikacji”, zamieszczając na serwerach zmienione pliki. Spróbuj zidentyfikować jakie wyzwania mogą się wiązać z taką zmianą uruchomionej aplikacji. Rozważ, jakie narzędzia mogą pomóc w publikacji nowej wersji aplikacji: *rsync*/*scp*, *(s)ftp*, własny skrypt, a może coś jeszcze innego?
4. [4 pkt.] Przygotuj (prosty, nie musi być super niezawodny ani elegancko zorganizowany) skrypt w dowolnym języku, który uruchomi nową instancję, połączy się z nią, zainstaluje na niej serwer HTTP i przygotuje jakąś minimalną (sam jeden *index.html* wystarczy) statyczną stronę, a następnie utworzy obraz takiego serwera, do późniejszego wykorzystania. Zweryfikuj, że nowe serwery, ręcznie uruchamiane z użyciem obrazu przygotowanego przez Twój skrypt, faktycznie od razu (bez dodatkowego konfigurowania) serwują tę prostą stronkę. Zastanów się, kiedy taka automatyzacja może być przydatna i jakie problemy rozwiązuje.

   Parę przydatnych linków:
   a. Dokumentacja operacji tworzenia instancji: https://cloud.google.com/compute/docs/instances/create-start-instance
   b. Dokumentacja operacji tworzenia obrazów: https://cloud.google.com/compute/docs/images/create-delete-deprecate-private-images
   c. Dostęp do GoogleComputeEngine za pomocą Pythona (warto spojrzeć w odnośnik “PyDoc reference”): https://developers.google.com/api-client-library/python/apis/compute/v1
   d. Dostęp do GoogleComputeEngine za pomocą Java (warto spojrzeć w odnośnik “JavaDoc reference”): https://developers.google.com/api-client-library/java/apis/compute/v1