Projektowanie i wdrażanie systemów w chmurze

Lista zadań na pracownię 2019.01.07

Zadania na tej pracowni są sformułowane z myślą o AWS, ale równie dobrze można je wykonać np. w GCP.

0. [0 pkt.] Sprawdź dane billingu kont chmurowych, których używasz. Zobacz, ile zostało creditsów i czy nie marnujesz ich na coś niespodziewanego.
1. [2 pkt.] Wykonując poniższe zadania, opisz infrastrukturę chmurową używając Terraforma.
2. [4 pkt.] Uruchom jakąś aplikację webową na kilku serwerach. Może to być jakieś gotowe narzędzie albo np. program z wcześniejszych pracowni. Zepnij te serwery grupą autoskalującą o **stałym** rozmiarze np. 2 lub 3. Przygotuj tej grupie taką *Launch Configuration*, by mogła uruchamiać nowe serwery identyczne do istniejących - będzie w tym celu niezbędne przygotowanie dobrego obrazu AMI. Skonfiguruj chmurowy load-balancer tak, by kierował ruch do serwerów w tej grupie autoskalującej[1].
   Skonfiguruj health-checki load-balancera i zweryfikuj, że działają poprawnie. Zadbaj, by grupa autoskalująca posługiwała się wynikami health-checków wykonywanych przez load-balancer. Upewnij się, że strona jest dostępna przez load balancer i że użytkownik nie widzi (przynajmniej na pierwszy rzut oka) który serwer obsłużył zapytanie.
   Sztucznie "zepsuj" jeden z serwerów - na przykład zabij w nim proces serwera HTTP. Upewnij się, czy wymiana serwera zajdzie całkiem automatycznie - czy grupa autoskalująca uzna serwer za *unhealthy*, czy wyłączy go i uruchomi nowy oraz czy ten nowy będzie od razu gotowy do pracy i zarejestrowany w load-balancerze. Sprawdź, czy podczas tego procesu strona **jest nieprzerwanie dostępna** z punktu widzenia jej użytkowników.
3. [5 pkt.] Wybierz jakieś zadanie kosztowne obliczeniowo. Może być to np. kompresja danych, jakieś zadanie kryptograficzne albo coś podobnego[2]. Jeżeli na innym przedmiocie pisałeś program, który wykonuje duże obliczenia, a łatwo je podzielić na części, to jest to świetna okazja, by uruchomić go w chmurze[3]. Najlepiej, by małemu serwerowi wykonanie pojedynczego zadania zajmowało tak co najmniej kilkadziesiąt sekund[4], przy pełnym wykorzystaniu CPU.
   Utwórz jedną kolejkę SQS. Przygotuj mały program/skrypt, który oczekuje parametrów zadania w wiadomości z kolejki i gdy otrzyma wiadomość z SQS, uruchamia wybrane przez Ciebie zadanie obliczeniowe. Niech wynik zostanie gdzieś zapisany, może np. zostać wysłany do S3. Przygotuj AMI serwera gotowego do przetwarzania takich zapytań. Zadbaj, aby wiadomość nie była kasowana z kolejki SQS, jeżeli obliczenia zostaną przerwane.
   Skonfiguruj grupę autoskalującą zarządzającą takimi workerami - nadaj jej zmienny rozmiar. Jakie jest sensowne minimum serwerów grupy w takim scenariuszu? Możesz użyć serwerów typu *EC2 Spot*, aby zaoszczędzić pieniądze. Zadbaj, by grupa samoczynnie dobierała liczbę serwerów tak, że gdy nie ma nic do roboty, to jest mało serwerów, a gdy przybywa zadań lub obliczeń, to grupa rośnie. Jakie metryki będziesz monitorować? Jak dobierzesz polityki skalowania do tego scenariusza?
   Przygotuj prosty skrypt (nie musisz go umieszczać na serwerze, użyj go lokalnie), który wysyła bardzo wiele wiadomości SQS zlecając obliczenia, które składają się na jakąś większą całość. Zaobserwuj, czy grupa skaluje się stabilnie i zgodnie z oczekiwaniami, a po zakończeniu obliczeń

---

[1] Jeżeli chcesz "wytestować" używanie domeny symbolicznej, a nie chcesz jej kupować, możesz użyć wpisu w lokalnym pliku */etc/hosts* na maszynie z której będziesz testować stronę.

[2] Odradzamy jednak łamanie sum kryptograficznych czy kopanie kryptowalut, gdyż takie obliczenia prawie zawsze naruszają regulamin usług chmurowych.

[3] A jeżeli zupełnie nie masz pomysłu, to napisz program, który rozwiązuje jakąś zagadkę brute-forcem, a jakiś parametr wejściowy określa mu, która część przestrzeni rozwiązań ma być przeszukana.

[4] Może być więcej, ale wtedy testowanie rozwiązania zajmie trochę więcej czasu. Jeśli natomiast będzie mniej niż kilkadziesiąt czasu, to zadbaj, by zlecić workerom wykonanie bardzo wielu takich małych tasków, by je porządnie obciążyć.

oszczędza pieniądze zmniejszając liczbę serwerów do minimum[5]. Przeanalizuj jak konfiguracja grupy wpływa na tempo przetwarzania zadań - mogą się przydać metryki SQS.

4. [1 pkt.] Dodatkowo przekonfiguruj zadanie 3 tak, aby pracowały dwie grupy autoskalujące używające różnych rodzajów serwerów (np. jedna *on-demand*, druga *spot*), a ich polityki skalowania nadawały większy priorytet skalowaniu jednej z nich. Upewnij się, że w normalnym przypadku skaluje się jedna grupa, a gdy z jakiś przyczyn już nie może, to wtedy druga uruchamia dodatkowe serwery.
5. * [1 pkt. extra] W zadaniu 3 wymyśl i zaimplementuj mechanizm, dzięki któremu nazwa kolejki i bucketa S3 nie będą trwale zaszyte w obrazie, tylko będzie można je konfigurować jako parametr grupy autoskalującej, razem z resztą infrastruktury chmurowej.

[5] Pomiędzy wykonaniem zadania a jego prezentacją na pracowni, możesz ręcznie przestawić min/desired/max grupy autoskalującej na 0/0/0.