Projektowanie i wdrażanie systemów w chmurze

Lista zadań na pracownię 2022.02.01

Maksimum liczone za tę listę to 8 - wszystkie punkty zdobyte ponadto liczą się jako ekstra. Wybór zadań/podpunktów jest dowolny.

1. **[razem 6 pkt]** Użyjemy rozwiązań serverless by zbudować prosty system regularnie archiwizujący dane z cudzej strony internetowej oraz przedstawiający statystyki z tych zebranych danych. Tematykę zbieranych danych, jak i dokładną stronę, z której będziesz je zbierać, możesz wybrać samodzielnie. Zadbaj tylko, by na zebranych danych dało się liczyć jakieś proste statystyki i prezentować je na wykresie[1].
   a. [2 pkt] Przygotuj funkcję chmurową, która będzie samoczynnie uruchamiać się co jakiś czas, pobierać wybraną stroną internetową i zapisywać wyciągniętą z niej wartość do bazy danych. Warto napisać tę funkcję tak, by dało się ją łatwo testować lokalnie, przed umieszczeniem w chmurze. Nie bój się parsować HTML strony, jeżeli to konieczne do wyciągnięcia obserwowanych danych[2]. Oznacz pomiar aktualną datą i godziną, i zapisz do bazy danych. Prawdopodobnie łatwo będzie użyć bazy typu datastore/bigtable, bo do tych baz funkcja chmurowa nie będzie potrzebować kontekstu sieciowego. Baza nie powinna być dostępna publicznie. Zadbaj, by liczby trafiające do bazy, były już oczyszczone i znajdowały się w formacie najwygodniejszym do liczenia statystyk. Na potrzeby testów, możesz skonfigurować tę funkcję tak, aby uruchamiała się nadmiernie często, na przykład raz na minutę. Oddając zadanie zadbaj, by baza ze zbieranymi statystykami nie była pusta, pozwól funkcji popracować co najmniej przez kilka godzin, a najlepiej dzień-dwa.
   b. [2 pkt] Skonfiguruj osobną funkcję chmurową, która będzie odpowiadać na requesty HTTP, przedstawiając statystyki wyciągnięte z bazy danych. Przygotuj co najmniej dwa endpointy liczące dwa różne rodzaje statystyk[3], każdy z nich powinien przyjmować argument jakoś wyrażający zakres czasu, z którego liczymy statystykę[4]. Jeśli chcesz, możesz dodać też endpoint, który w odpowiedzi wysyła gotowy obrazek z wykresem wartości we wskazanym przedziale czasu - przygotowanie wykresu tutaj może nieco ułatwić trzeci podpunkt. Ustaw lub znajdź publiczny URL tej funkcji i upewnij się, że odpowiada na zewnętrzne requesty.
   c. [2 pkt] Przygotuj prosty frontend - stronkę internetową, która umożliwia użytkownikom wgląd w te statystyki. Ta stronka powinna składać się **wyłącznie z treści statycznej** (dzięki czemu nie będzie potrzebować serwera). Wśród tej treści statycznej powinny znajdować się odniesienia do endpointów funkcji z poprzedniego podpunktu, które dostarczą aktualne dane do frontendu. Zamieść na stronie prosty skrypt (może to być surowy JS), który będzie odpytywać te endpointy o statystykę i przedział czasu wybrane przez użytkownika i prezentować je w postaci jakiegoś wykresu. Jeżeli generowanie wykresu prosto w przeglądarce użytkownika okaże się trudne w wybranej technologii, możesz zastąpić go statycznym odniesieniem do obrazka-wykresu dynamicznie generowanego przez funkcję z podpunktu *b*. Opublikuj stronę nie przygotowując serwerów, za pomocą opcji “static website hosting” w Google Cloud Storage.

---

[1] Praktycznie każda wyrażona liczbą dynamiczna wartość jaką zobaczysz gdziekolwiek w internecie nadaje się do tego zadania, ale jeśli nie masz pomysłu, to rozważ np. wybrany kurs waluty, notowania na giełdzie, dane meteorologiczne, nagłówki wiadomości na portalu z newsami, liczba subskrypcji kanału na Youtube lub innych mediach społecznościowych, pieniądze zebrane przez jakąś zbiórkę, popularność fraz na Twitterze itp...

[2] Jeżeli używasz Pythona, polecamy bibliotekę Beautiful Soup, którą można błyskawiczne znaleźć interesujący nas wpis w dokumencie HTML. Możesz też przygotować wyrażenie regularne, które wyciągnie szukaną wartość z treści strony.

[3] Np. średnia wartość w zadanym okresie czasu albo wariancja, albo mediana, albo regresja liniowa - sensowne statystyki zależą od rodzaju zbieranych danych.

[4] Może to być przedział dat/godzin od-do albo np. “ostatnie N godzin”.

2. **[razem 5 pkt]** Wyobraźmy sobie system składający się z wielu instancji serwerów, który używany jest tylko w godzinach pracy biura. Aby oszczędzić wydatki, chcielibyśmy automatycznie wyłączać te serwery wieczorem, oraz włączać je ponownie rano. Temu mechanizmowi powinny podlegać nie wszystkie instancje w projekcie, tylko te, i wszystkie te, które są oznaczone konkretną etykietą / tagiem.
   a. [2 pkt] Przygotuj funkcje chmurową w stylu serverless, która będzie automatycznie wywoływana wg. wybranego harmonogramu. Niechaj o niektórych wskazanych godzinach wyłącza wszystkie instancje z pasującą etykietą, a o innych godzinach włącza je z powrotem. Skonfiguruj harmonogram uruchamiania funkcji chmurowej tak, aby serwery pracowały tylko w dni robocze, w godzinach 8-16. Zadbaj o to, by zmiana ustawień harmonogramu była łatwa do wdrożenia - np. używając Terraforma połącz parametry harmonogramu w jedną zmienną z listą wydarzeń, na podstawie której generowane będą odpowiednie zasoby.
   b. [2 pkt] Przypuśćmy, że czasami pracownicy potrzebują aktywnych serwerów w weekend lub wieczorami. Przygotuj endpoint HTTP który umożliwi uruchomienie tych samych serwerów na żądanie, bez dostępu do konta chmurowego, jedynie wysyłając request HTTP pod odpowiedni URL. Zabezpiecz endpoint prostym hasłem. Możesz użyć tej samej funkcji chmurowej, lub zbudować drugą osobną, ale w obu przypadkach ogranicz podwajanie kodu, wykorzystując ten sam codebase do obu celów.
   c. [1 pkt] Zaimplementuj wysyłkę powiadomienia przy każdym uruchomieniu tej funkcji, aby administrator systemu miał na oku wszystkie uruchomienia/wyłączenia serwerów za pomocą tej automatyzacji. Może to być np. wiadomość email, SMS, powiadomienie na Discordzie, itp. W treści **koniecznie** musi się znaleźć liczba serwerów, które zostały dotknięte zmianą stanu.