Wirtualne Sieci Obliczeniowe

Bartłomiej Krawczyk, Mateusz Brzozowski

Automatyzacja: Skalowalność i wysoka dostępność - serwisy bezstanowe

Skalowalność i wysoka dostępność

Funkcje:

- automatyzacja skalowania usługi (dodawanie/usuwanie VM),
- równoważenie obciążenia (np. haproxy)
- zwiększanie niezawodności.

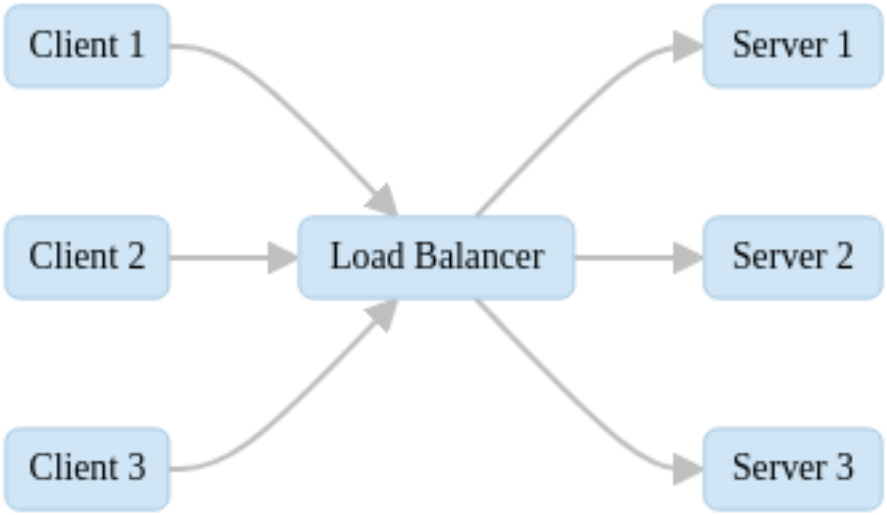
Sprawdzić jaki wpływ na obsługę ma awaria jednej/większej liczby maszyn. Serwis bezstanowy.

Automatyzacja zarządzania maszynami wirtualnymi:

- Zestaw skryptów/program ułatwiający zarządzanie VM/klastrem
- Opracować odpowiedni scenariusz
- Użycie narzędzi!

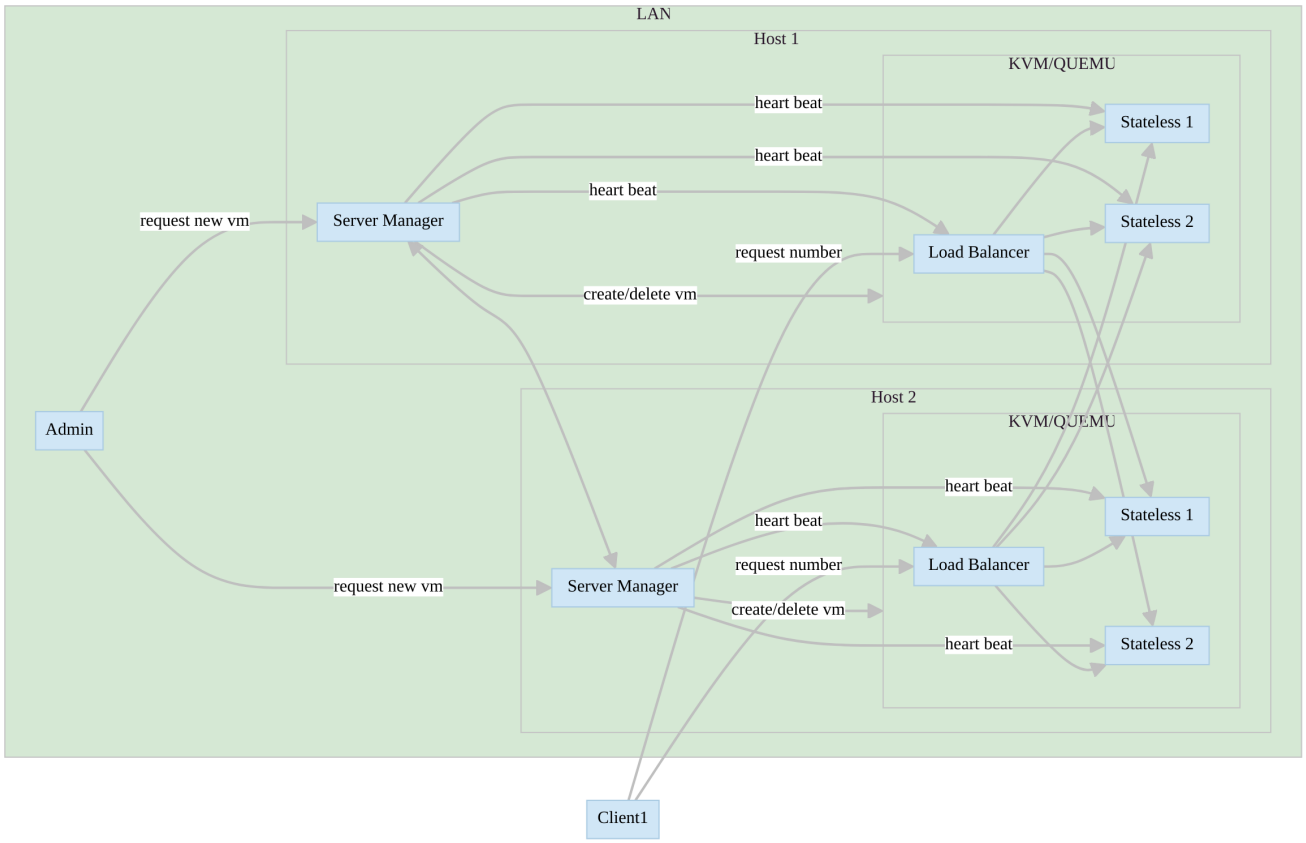
Scenariusz

- Wiele serwisów bezstanowych, które zwracają losową wartość (bool, int, float, double),
- Cały ruch użytkowników przechodzi przez load balancer (nginx), który jest uruchomiony na oddzielnej maszynie wirtualnej,
- Jeden load balancer przypisany jest do jednego menadżera,
- Jeden menadżer zarządza całym klastrem i jest uruchomiony lokalnie na fizycznej maszynie (nie na VM),



- Na każdym laptopie w naszym klastrze uruchomiony jest daemon manager, który zarządza maszynami wirtualnymi,
- Każdy manager posiada endpointy do utworzenia i usuwania maszyny wirtualnej z serwisem bezstanowym, do aktualizacji serwisów z innych menadżerów,
- Menadżerzy znają wzajemnie swoje adresy IP, propagują między sobą informacje o uruchomionych maszynach,

- Każdy menadżer posiada przypisaną pulę dostępnych lokalnych adresów IP, tak aby unikać kolizji z innymi menadżerami.



Ryzyko

- Maszyna / serwis z serwisem bezstanowym umiera:

Menadżer przez cały czas działa serwisu utrzymuje połączenie heartbeat, serwis co jakiś czas wysyła wiadomość zwrotną o treści: `data: {"status": "OK"}` sygnalizującą poprawne działanie serwisu. Jeśli menadżer nie wykryje przez określony czas połączenia, kilkukrotnie próbuje nawiązanie połączenia, jeśli się to nie uda to usuwamy taką maszynę i menadżer stawia nową maszynę w jej miejsce.

Adres IP nowej maszyny pozostaje taki sam jak adres usuniętej maszyny. W przypadku błędnej odpowiedzi serwisu, load balancer (nginx) automatycznie przekierowuje żądanie do innej dostępnej maszyny, zgodnie z konfiguracją opcji proxy_next_upstream.

- Maszyna / serwis z load balancerem umiera:

Heartbeat w ramach load balancera działa podobnie jak w serwisie bezstanowym.

Mamy jeden publiczny adres ip, który jest na starcie przypisany do jednego menadżera, jeśli menadżer ma problem ze swoim load balancerem, to mianuje drugiego menadżera głównym i przypisuje do niego publiczny adres ip, a nasz load balancer wyłączamy i próbujemy postawić na nowo z innym adresem ip.

- Serwis z managerem umiera:

Coś bardzo złego się dzieje na naszym serwerze. Jako, że jest to projekt studencki zakładamy w takich wypadkach interwencję administratora. Zamiast zabezpeczać się przed tymi sytuacjami.

- Kolizja adresów ip:

Każdy manager ma przypisaną własną unikalną pulę adresów ip do przypisania do maszyn.

Sprzet

Dwa laptopy z systemem Ubuntu w tej samej sieci wifi.

Narzędzia

- Wirtualizator: KVM
- Obraz: Alpine Linux - Virtual
- Serwisy: Spring w kotlinie
- Zarządzanie konfiguracją maszyn: Ansible
- Skrypty testowe: Bash

Plan testów

Skrypt który odpytuje load balancer, działa z pewnym opóźnieniem, cały czas dopóki nie zostanie zatrzymany. W oddzielnej konsoli wyłączymy maszynę bezstanową/load balancer, podobnie na drugim laptopie i monitorujemy jak zachowuje się serwis, czy wszystkie odpowiedzi zwracane są poprawne.