Projektowanie i wdrażanie systemów w chmurze

Lista zadań na pracownię 2018.10.29

Będziemy przygotowywać środowisko złożone z kilku serwerów HTTP, podłączonych do wspólnego load balancera. Przygotuj konfigurację serwerów opisanych poniżej za pomocą Ansible[1]. Upewnij się, że konfiguracja jest kompletna, tzn. nie wymaga **żadnych** ręcznych akcji na serwerach (tak, żeby łatwo było wymienić serwer i automatycznie skonfigurować go od nowa).
Wyróżniamy trzy rodzaje serwerów w środowisku:

a. Serwery aplikacji (z serwerem HTTP)
b. Load-balancer (przyjmujący zapytania HTTP i rozdzielający je na serwery aplikacji)
c. Tzw. *Bastion host* (użytkownicy będą go używać jako pośrednika przy dostępie do pozostałych serwerów)

Planujemy używać jednego load-balancera i jednego *bastion hosta*, ale serwerów aplikacji powinno być więcej, np. trzy. Przygotuj playbooki dla każdego rodzaju serwera. **Użyj ról Ansible**, aby opisać grupy zadań, a następnie wykorzystaj je w playbookach. Tworząc playbooki **upewnij się**, że działają poprawnie nie tylko umieszczając konfigurację na świeżym, czystym serwerze, ale także że nadają się do wprowadzania drobnych zmian w konfiguracji na serwerze poprzez ponowne wykonanie tego samego playbooka. Warto wykorzystać tagi do zadań.
Serwery aplikacji powinne być pozbawione publicznego adresu IP, ale (dla wygody) na czas konfigurowania mogą tymczasowo posiadać publiczne IP, które na koniec wyłączamy[2].
Dla wygody zalecamy przygotować plik *inventory*, w którym znajdą się adresy IP maszyn, które konfigurujemy.
Oczekiwana konfiguracja serwerów powinna wyglądać jak następuje. Omówione poniżej cechy serwerów są w sporym stopniu niezależne, jak najbardziej dopuszczamy wykonanie tylko części z nich.

1. [2 pkt.] Serwery aplikacji powinny mieć uruchomiony serwer HTTP (np. *nginx* lub *apache*)[3] udostępniający jakąś prostą witrynę - może być całkiem banalna.
2. [2 pkt.] Load-balancer powinien mieć uruchomione *haproxy* (lub *nginx* w trybie reverse-proxy) oraz być skonfigurowany tak, by przekazywał zapytania HTTP do serwerów aplikacji. W tym celu konfiguracja load-balancera może potrzebować adresów IP serwerów aplikacji, a ponieważ te mogą się zmieniać, przekaż je w konfiguracji jako zmienne (ustawiane w playbooku, inventory albo z linii komend przy wykonywaniu playbooka).
3. [2 pkt.] Wszystkie serwery powinny odmawiać logowania SSH za pomocą hasła (używamy tylko par kluczy). Do tego celu może przydać się opcja *PasswordAuthentication* w pliku */etc/ssh/sshd_config* oraz moduł ansible *replace*, który pozwala wygodnie nadpisać drobny fragment pliku.
   Dodatkowo wszystkie serwery poza *bastion host*em muszą odmawiać połączeń SSH przychodzących z publicznego internetu i zezwalać tylko na połączenia przychodzące z sieci lokalnej - w ten sposób aby połączyć się z serwerem aplikacji trzeba najpierw zalogować się na *bastion host*, a następnie stamtąd zalogować się na właściwy serwer. W tym celu można przestawić *ListenAddress* serwera SSH, lub ustawić odpowiednie reguły zapory ogniowej.
4. [2 pkt.] Na wszystkich serwerach powinni być dostępni użytkownicy Alice i Bob, którzy będą się logować zdalnie na serwery aplikacji. Nie powinni mieć uprawnień administratora, ale na serwerach aplikacji powinni mieć prawa do zamieszczania i edycji plików strony internetowej. Być może w tym celu wygodnie jest skonfigurować odpowiednią grupę użytkowników. Pamiętaj, że serwery nie

[1] Jeżeli chcesz, możesz użyć innego narzędzia (Chef, Puppet, Salt), ale zorganizuj wtedy swój kod tak, by wykorzystać mechanizmy analogiczne do tych, które będziemy zalecać w treści zadania.
[2] W prawdziwej konfiguracji dobrym zwyczajem jest, by serwery aplikacji nigdy nie były dostępne z publicznego internetu, a wszystkie połączenia użytkowników do serwerów aplikacji odbywały się przez *bastion hosta*. W ramach tej pracowni pozwalamy tymczasowo złamać ten zwyczaj, jeżeli ułatwi to realizację niektórych zadań.
[3] Jeżeli chcemy użyć własnego skryptu/programu jako serwera HTTP, to musi on być zainstalowany jako usługa systemowa.

powinny umożliwiać logowania za pomocą hasła, musisz zatem zamieścić na serwerze publiczne części kluczy SSH obu użytkowników (za pomocą Ansible, nie np. konsoli GCP).

5. [2 pkt.] Musimy zbierać logi serwerów HTTP w bezpieczne miejsce, poza serwerami aplikacji. Istnieją specjalne programy, które zajmują się tym zadaniem (można ich użyć), ale wystarczy nam dużo prostsze rozwiązanie - wystarczy *cron job* który raz na kilka minut będzie wysyłał pliki logów serwera HTTP np. do S3 lub Cloud Storage, albo inne bezpieczne miejsce.
   Do tego, na wszystkich serwerach (zwłaszcza na *bastion hoscie*) musimy też zbierać logi kto kiedy zalogował się na serwerze[4]. Można napisać własny skrypt, który będzie zbierał logi, ale musi być uruchomiony jako usługa systemowa.
6. [2 pkt.] Serwery aplikacji powinny mieć uruchomiony serwer FTP, skonfigurowany tak, by prezentował (i umożliwiał zamieszanie) pliki strony internetowej. Alice i Bob będą używać FTP aby publikować nową wersję strony. Upewnij się że mają dostęp do serwera i że zamieszczone pliki stają się dostępne przez HTTP[5].

Przykłady, które pokazaliśmy na wykładzie dostępne są tutaj: https://github.com/rafalcieslak/cloud18. W plikach *using_roles{,2}.yml* znajduje się krótki komentarz jak używać roli w Ansible - bardzo prosty sposób na posprzątanie kodu wspólnego między wieloma playbookami.
Dokumentacja modułów Ansible: http://docs.ansible.com/ansible/latest/modules_by_category.html
Szczegóły jak przygotować *playbook*: http://docs.ansible.com/ansible/latest/playbooks.html
Szczegóły o tworzeniu ról: http://docs.ansible.com/ansible/latest/playbooks_reuse_roles.html

[4] W zależności od dystrybucji, jest to zazwyczaj zapisywane do pliku */var/log/auth.log* lub */var/log/secure*, wystarczy że zachowamy gdzieś cały ten plik.
[5] Prawdę mówiąc, takie rozwiązanie nie jest praktyczne, gdy dane strony trzeba umieszczać na wielu serwerach aplikacji. Ale skonfigurowanie serwera FTP bywa nieoczywiste.