Wstęp do wirtualizacji

Hello

Łukasz Pieczonka

DevOps @ LU-MEDIA

Agenda

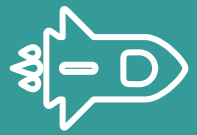
1. Wprowadzenie do wirtualizacji
   Czym jest wirtualizacja, co można wirtualizować,
   przykłady hipernadzorców, typy wirtualizacji maszyn
2. Przygotowanie do pracy
   Instalacja Virtualbox i Vagrant
3. Wprowadzenie do Vagrant
   Wspólne opracowanie pliku vagrantfile - obraz bazowy oraz
   przekierowanie 2 dowolnych portów
4. Praca samodzielna, z dokumentacją - celem jest przygotowanie
   skryptów instalacyjnych dla Dockera, docker-compose oraz innych
   zależności (cURL, wget, git..) oraz dołączenie ich do vagrantfile

Historia: baremetal

Problemy:

- Powolne wdrażanie
- Duże koszta (licencje/sprzęt/utrzymanie)
- Problemy ze skalowaniem
- Trudne w migracji

1. Wprowadznie do wirtualizacji



Czym jest wirtualizacja

Wirtualizacja polega na utworzeniu symulowanego (wirtualnego) środowiska komputerowego przez oprogramowanie istnienia zasobów logicznych, które wykorzystują ustalone podczas konfiguracji zasoby fizyczne. Obecnie wirtualizować możemy niemal wszystko co związane z IT Najpopularniejsze to całe systemy operacyjne serwerowe, desktopowe wirtualizuje się również aplikacje, pulpity sieci komputerowe oraz pamięci masowe.

Co można wirtualizować ?

- Serwery
- Sieci
- Pamięci masowe
- Oprogramowanie

Hipernadzorca

hiperwizor, od ang. hypervisor

narzędzie niezbędne do zarządzania procesami wirtualizacji. Pojęcie hipernadzorca (hiperwizor) pochodzi od nazwy supervisor, czyli nadzorca – programu kontrolującego pracę komputera.

Hipernadzorca

opis działania

Ponieważ wirtualizator pozwala maksymalnej ilości procesów wirtualnego systemu operacyjnego wykonywać swoje instrukcje bezpośrednio na zasobach sprzętowych, niezbędny jest system kontroli. Jeżeli określona operacja zwraca błąd ochrony (nie daje się z jakiejś przyczyny wykonać bezpośrednio na danym zasobie sprzętowym), jest przechwytywana i emulowana przez hipernadzorcę. Hipernadzorca decyduje, które procesy zwirtualizowanego systemu operacyjnego można wykonywać bezpośrednio na zasobach sprzętowych, a które należy emulować. W niektórych systemach operacyjnych hipernadzorca jest nazywany menedżerem maszyn wirtualnych (ang. virtual machine manager).

Hipernadzorca

Dodatkową funkcją, jaką pełni hipernadzorca jest pośredniczenie w przekazywaniu przerwań pomiędzy wirtualnym systemem a zasobami sprzętowymi i ewentualna emulacja urządzenia po przyjęciu przerwania.

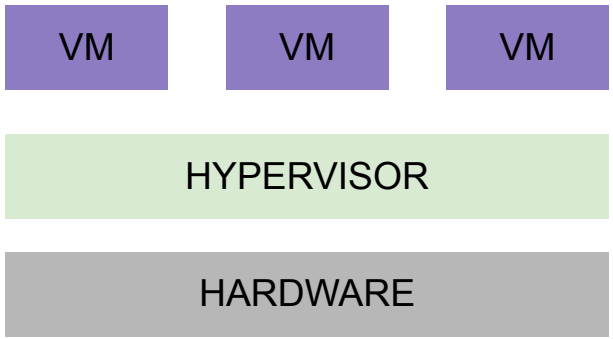
Przykłady hipernadzorców

- Citrix XenServer
- XCP-ng
- KVM
- ESXi VMWare
- Microsoft Hyper-V
- Microsoft Virutal PC
- Virtualbox

Klasyfikacja

typy wirtualizacji

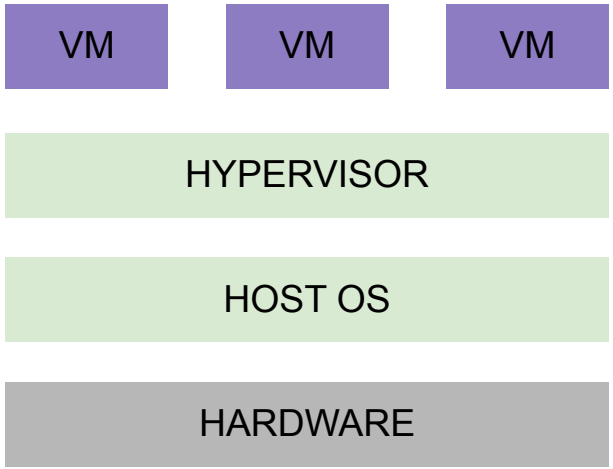
Wirtualizacja typu pierwszego tzw. Bare Metal Hypervisor pracuje bezpośrednio na sprzęcie komputera fizycznego np. Xen Server (XCP-NG), VMware ESXi, Microsoft Hyper-V co przekłada się najwyższą wydajność i efektywność, a także na bezpieczeństwo. Hipernadzorcy typu 1 bazują na pełnej wirtualizacji z akceleracją sprzętową.



Klasyfikacja

typy wirtualizacji

Wirtualizacja typu drugiego - hypervisor działa w systemie operacyjnym gospodarza i poprzez system operacyjny przydziela zasoby dla maszyn wirtualnych np. Virtualbox instalowane w istniejącym systemie operacyjnym jako dodatkowa aplikacja. Ciekawym przypadkiem jest wirtualizator KVM który instalujemy jako aplikacja ale integracja z jądrem Linuxa sprawia żę funkcjonuje w rzeczywistości jako typ 1



Klasyfikacja

Wirtualne dyski twarde

Przechowywanie jest to dość obszerny temat i wygląda różnie w zależności od hypervizora.

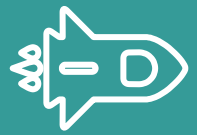
Wirtualny dysk twardy - jest to dysk na którym są przechowywane te same pliki i katalogi, które znajdują się fizycznym dysku twardym, fizycznie jest to plik/pliki.

Przykłady typów dysków wirtualnych:

- VDI (VirtualBox Disk Image)
- VHD (Virtual Hard Disk)
- VMDK (Virtual Machine Disk)
- QCOW (QEMU Copy-On-Write)
- QED (QEMU enhanced disk)

Rozmiar może być ustawiony na ustalony rozmiar lub być skonfigurowany tak, aby rosnąć wraz z upływem czasu.

2. Przygotowanie do pracy



Systemem Ubuntu 18.04 / 20.04
Oprogramowanie:

- Visual Studio Code
- Git
- Vagrant
- Virtualbox

Instalacja Virtualbox

https://www.virtualbox.org/wiki/Downloads

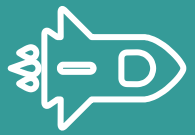
Instalacja Vagrant by HashiCorp

HashiCorp
Vagrant

https://www.vagrantup.com/downloads

3. Wprowadzenie do Vagrant

czyli jak łatwo stworzyć środowisko dla pracy/testów



Vagrant to narzędzie, które umożliwia szybkie tworzenie maszyn wirtualnych na podstawie plików konfiguracyjnych Vagrantfile. Korzystanie z tego narzędzia jest bardzo proste i szybkie, zasadniczo wykorzystujemy jeden plik konfiguracyjny do stworzenia całego systemu, możemy skonfigurować sieć i usługi oraz wiele innych parametrów. Cała magia zaczyna się od wydania polecenia vagrant up

Vagrant boxes

Vagrant korzysta z tzw. boxów, czyli wcześniej przygotowanych obrazów maszyn

```
vagrant box add ubuntu/focal64 --provider virtualbox
vagrant box list
vagrant box remove
```

Vagrant

Przydatne polecenia i komendy

- `vagrant init [name [url]]` - inicjuje bieżący katalog jako środowisko Vagrant poprzez utworzenie pliku *Vagrantfile*, jeśli jeszcze nie istnieje.
- `vagrant up [name|id]` - tworzy i konfiguruje maszynę (maszyny) na podstawie *Vagrantfile*
- `vagrant ssh [name|id] [-- extra_ssh_args]` - loguje się do działającej maszyny za pomocą SSH
- `vagrant status [name|id]` - podaje status maszyny
- `vagrant global-status` - informuje o stanie wszystkich aktywnych środowisk Vagrant w systemie dla aktualnie zalogowanego usera
- `vagrant suspend [name|id]` - wstrzymuje maszynę, ale jej nie wyłącza (`halt`) ani nie niszczy (`destroy`)
- `vagrant halt [name|id]` - wyłącza uruchomioną maszynę
- `vagrant destroy [name|id]` - zatrzymuje uruchomioną maszynę i niszczy wszystkie zasoby, które zostały utworzone podczas jej tworzenia
- `vagrant resume [name|id]` - wznawia maszynę, która była wcześniej zawieszona, np. przez polecenie `suspend`

Wspólne opracowanie pliku vagrantfile

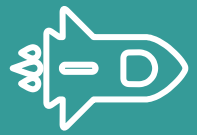
Inicjalizacja środowiska z ubuntu

```
vagrant init ubuntu/focal64
```

Wspólne opracowanie pliku vagrantfile

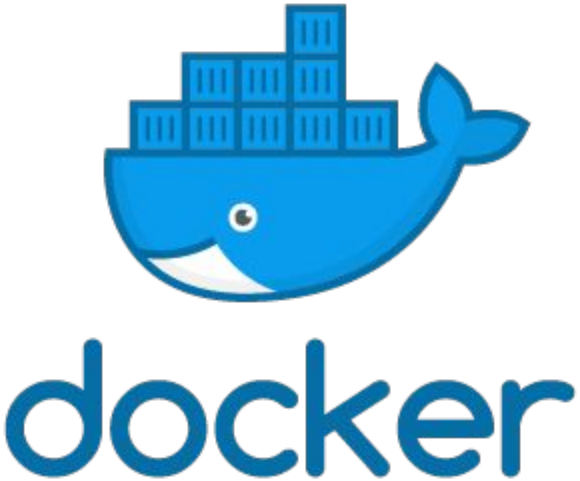
```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"
  # config.vm.box_check_update = false
  config.vm.network "forwarded_port", guest: 80, host: 10080
  # config.vm.network "forwarded_port", guest: 80, host: 8080, host_ip: "127.0.0.1"
  config.vm.network "private_network", ip: "192.168.33.10"
  # config.vm.network "public_network"
  config.vm.synced_folder "../data", "/vagrant_data"
  config.vm.provider "virtualbox" do |vb|
    vb.name = "vm-1"
    vb.memory = "512"
  end
  config.vm.provision "shell", inline: <<-SHELL
    apt-get update
    apt-get install -y apache2
  SHELL
end
```

4. Praca samodzielna



Przygotowanie skryptów instalacyjnych dla Dockera, docker-compose oraz innych zależności (cURL, wget, git..) oraz dołączenie ich do vagrantfile

Install Docker Engine on Ubuntu



https://docs.docker.com/engine/install/ubuntu/

Dzieki

You can find me at
@username & user@mail.me