
# Vagrant w 10 minut

autor: Grzegorz Adamczyk

# Czym jest Vagrant?

**Vagrant to narzędzie, które pozwala zautomatyzować zarządzanie środowiskami wirtualnymi.**

Umożliwia tworzenie maszyn wirtualnych na podstawie pliku konfiguracyjnego. Sam nie dostarcza mechanizmów wirtualizacyjnych, wymaga do działania zewnętrznych dostawców (providerów). Domyślnym i najpopulrniejszym jest VirtualBox.

# Do czego można wykorzystać Vagrant

**Świetnie nadaje się do tworzenia, zarządzania i dystrybucji przenośnych środowisk developerskich.**

- Używanie kilku środowisk developerskich na jednej maszynie fizycznej tworzy wiele problemów. Zamiast instalować wszystkie aplikacje na jednej maszynie fizycznej, lepiej użyć maszyn wirtualnych. Vagrant ułatwia konfigurowanie środowisk złożonych z wielu maszyn wirtualnych
- Raz stworzoną konfigurację maszyn można łatwo uruchomić na dowolnej maszynie fizycznej na której jest zainstalowany Vagrant

# Dlaczego Vagrant

- **Skalowalność**: możemy tworzyć zarówno pojedyncze maszyny jak i złożone środowiska
- **Automatyzacja i przyspieszenie tworzenia maszyn wirtualnych**: Vagrant oferuje gotowe obrazy zamiast instalacji systemu, raz pobrany obraz przechowywany jest w lokalnym repozytorium i może być wykorzystywany wielokrotnie bez konieczności ponownego pobierania
- **Gotowe obrazy**: duży wybór obrazów dla domyślnego providera (VirtualBox)
- **Wsparcie dla różnych dostawców wirtualizacji**: Vagrant potrafi współpracować z VirtualBox, HyperV, vSphere, Docker, Xen, Aws ....

# Dlaczego Vagrant

- **Niska bariera wejścia**: minimalny plik konfiguracyjny to 3 linijki, maszyna zbudowana według ustawień domyślnych posiada możliwość zalogowania po ssh, dostęp do internetu oraz współdzielony folder do wymiany plików z hostem gospodarza
- **Podejście Infrastructure As a Code**: pracujemy z tekstowym plikiem konfiguracyjnym zamiast z GUI i myszką. Raz stworzoną konfigurację można wykorzystywać wielokrotnie, plikami można zarządzać przy pomocy systemu kontroli wersji (np. git)
- **Współpraca z różnymi systemami operacyjnymi**: Vagrant działa na Windows, Linux, OS X

# Jak stworzyć i uruchomić maszynę wirtualną?

## Tworzymy plik konfiguracyjny w którym definiujemy:

- **system operacyjny, który chcemy uruchomić**: wskazujemy nazwę tzw. pudełka
- **konfigurację sprzętową maszyny**: procesor, pamięć, dyski (to na co pozwala dany provider)
- **konfigurację sieci**
- **provisioning**: jakie polecenie lub skrypt ma zostać uruchomiony po starcie maszyny witrualnej. Obsługiwane są również takie narzędzia jak Ansible, Pupet, Salt

Jeżeli nie wskażemy konkretnej konfiguracji, zostaną użyte wartości domyślne. Jedyne ustawienie, którego nie można pominąć to nazwa pudełka

# Pudełka (Boxes)

Pudełko w Vagrancie to gotowy do uruchomienia obraz systemu operacyjnego

## Skąd wziąć pudełka?

Potrzebujemy tylko nazwę pudełka, Vagrant sam pobierze wszystkie potrzebne pliki

Dostępne pudełka możemy wyszukać na stronie Vagrant Cloud:
https://app.vagrantup.com/boxes/search

HashiCorp Vagrant Cloud

Search Pricing Vagrant Help Create an Account Sign In

# generic / ubuntu2204 Vagrant box

How to use this box with Vagrant:

Vagrantfile New

```
Vagrant.configure("2") do |config|
  config.vm.box = "generic/ubuntu2204"
end
```

## v4.2.16 currently released version

This version was created 3 months ago.

A build environment for use in cross platform development.

5 providers for this version.

- parallels Hosted by Vagrant Cloud (1.98 GB)
- libvirt Hosted by Vagrant Cloud (1.5 GB)
- virtualbox Hosted by Vagrant Cloud (1.44 GB)
- vmware_desktop Hosted by Vagrant Cloud (1.47 GB)
- hyperv Hosted by Vagrant Cloud (1.44 GB)

# Co wtedy gdy nie ma pudełka takiego jak chcemy?

- Vagrant daje możliwość tworzenia własnych pudełek
- Udostępnia w tym celu narzędzie Packer: https://www.packer.io/
- Gotowe pudełko możemy umieścić na stronie Vagrant Cloud tak, aby mogli z niego inni

# Tworzymy pierwszą maszynę

## najprostszy plik konfiguracyjny

```
Vagrant.configure("2") do |config|
  config.vm.box = "debian/bookworm64"
end
```

Do tworzenia plików konfiguracyjnych vagranta używana jest składnia języka Ruby

# Domyślny plik konfiguracyjny

## Alternatywa do budowania własnego pliku od podstaw

Kiedy chcemy skorzystać z gotowego szablonu pliku konfiguracyjnego, możemy wykorzystać polecenie `vagrant init`

Polecenie stworzy plik o nazwie Vagrantfile w bieżącym katalogu. Plik zawiera przykładową konfigurację zabezpieczoną przed wykonaniem komentarzami. Aby skorzystać z wybranych ustawień, należy usunąć komentarze z początków linii.

# Uruchamiamy pierwszą maszynę

Przechodzimy do katalogu z naszym plikiem Vagrantfile i uruchamiamy maszynę:

```
vagrant up
```

```
PS C:\VAGRANT\default> vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'debian/bookworm64' version '12.20230615.1' is up to date...
==> default: Resuming suspended VM...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
==> default: Machine booted and ready!
==> default: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> default: flag to force provisioning. Provisioners marked to run always will still run.

==> default: Machine 'default' has a post `vagrant up` message. This is a message
==> default: from the creator of the Vagrantfile, and not from Vagrant itself:
==> default:
==> default: Vanilla Debian box. See https://app.vagrantup.com/debian for help and bug reports
PS C:\VAGRANT\default>
```

# Sprawdzamy czy maszyna uruchomiła się

```
vagrant status
```

```
PS C:\VAGRANT\default> vagrant status
Current machine states:

default                   running (virtualbox)

The VM is running. To stop this VM, you can run `vagrant halt` to
shut it down forcefully, or you can run `vagrant suspend` to simply
suspend the virtual machine. In either case, to restart it again,
simply run `vagrant up`.
```

# Szczegółowe informacje o wszystkich maszynach

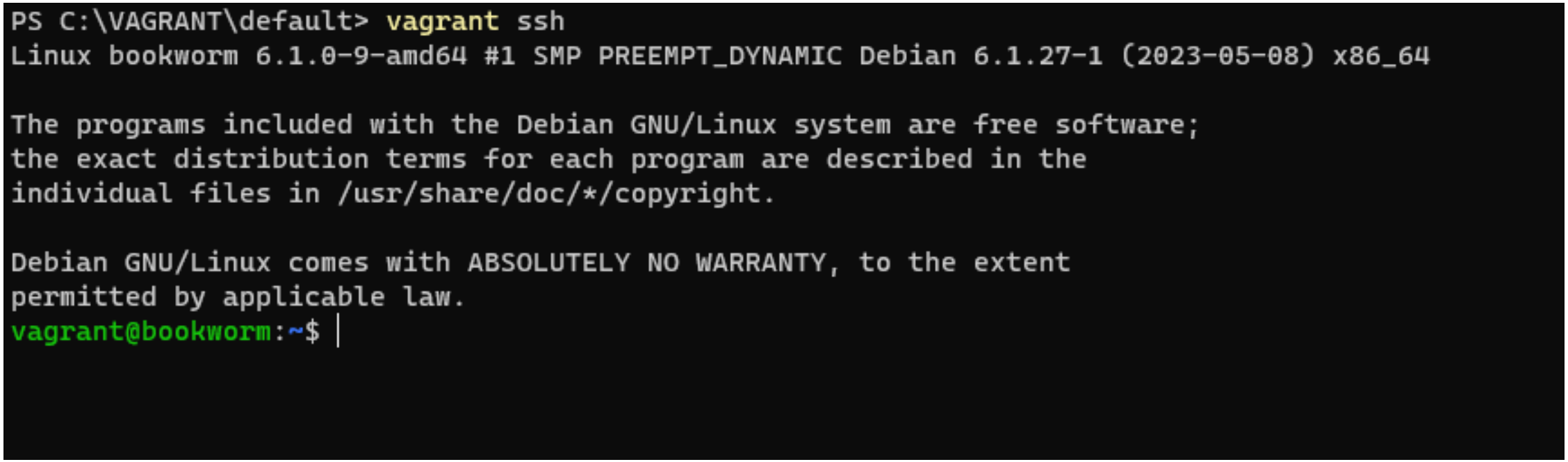
```
vagrant global-status
```

```
PS C:\VAGRANT\default> vagrant global-status
id       name    provider   state   directory
--------------------------------------------------------------------
70d3851  default virtualbox running C:/VAGRANT/default

The above shows information about all known Vagrant environments
on this machine. This data is cached and may not be completely
up-to-date (use "vagrant global-status --prune" to prune invalid
entries). To interact with any of the machines, you can go to that
directory and run Vagrant, or you can use the ID directly with
Vagrant commands from any directory. For example:
"vagrant destroy 1a2b3c4d"
```

# Logujemy się do maszyny

```
vagrant ssh
```



# Testujemy internet

W domyślnej konfiguracji maszyna wirtualna posiada dostęp do internetu

```
ping wp.pl
```

```
vagrant@bookworm:~$ ping wp.pl
PING wp.pl (212.77.98.9) 56(84) bytes of data.
64 bytes from www.wp.pl (212.77.98.9): icmp_seq=1 ttl=57 time=24.9 ms
64 bytes from www.wp.pl (212.77.98.9): icmp_seq=2 ttl=57 time=23.7 ms
64 bytes from www.wp.pl (212.77.98.9): icmp_seq=3 ttl=57 time=18.9 ms
64 bytes from www.wp.pl (212.77.98.9): icmp_seq=4 ttl=57 time=19.7 ms
64 bytes from www.wp.pl (212.77.98.9): icmp_seq=5 ttl=57 time=18.7 ms
64 bytes from www.wp.pl (212.77.98.9): icmp_seq=6 ttl=57 time=19.8 ms
64 bytes from www.wp.pl (212.77.98.9): icmp_seq=7 ttl=57 time=21.6 ms
64 bytes from www.wp.pl (212.77.98.9): icmp_seq=8 ttl=57 time=20.2 ms
64 bytes from www.wp.pl (212.77.98.9): icmp_seq=9 ttl=57 time=18.9 ms
64 bytes from www.wp.pl (212.77.98.9): icmp_seq=10 ttl=57 time=20.3 ms
```

# Sieć

Jeżeli używamy VirtualBox jako providera, domyślnie po stronie VirtualBox tworzona jest sieć typu NAT

- Maszyna ma adres ip
- Maszyna ma dostęp do internetu

# Testujemy folder współdzielony z hostem gospodarza

Bieżący folder z konfiguracją Vagranta jest domyślnie mapowany wewnątrz maszyny wirtualnej na folder /vagrant
Dzięki temu możemy w prosy sposób wymieniać pliki między hostem gospodarza a maszyną wirtualną

```
ls /vagrant
```

# Kończymy pracę z maszyną

## Wychodzimy z sesji ssh

```
exit
```

## Zatrzymujemy maszynę

```
vagrant halt
```

## Niszczymy maszynę

```
vagrant destroy
```