



Vagrant

[Введение](#)

[Vagrant](#)

[Установка Vagrant в Windows 10](#)

[Установка Vagrant в Linux \(Ubuntu 20.04\)](#)

[Создание виртуальной машины в Vagrant](#)

[Основные команды Vagrant](#)

[Подключение к виртуальной машине через SSH](#)

[Использование прокси для работы Vagrant](#)

[Заключение](#)

[Список использованных источников](#)

Введение

Разработка web-приложений начинается с настройки окружения, в котором будет запускаться и тестироваться код. Необходима среда разработки, в которой, к примеру установлена операционная система Linux со специфичными настройками, зависимости проекта и необходимые программы (MySQL, Nginx, PHP и т.д.).

От проекта к проекту требования могут меняться. Разработчик может работать над несколькими проектами одновременно и у каждого из них будут разные зависимости. Так же над каждым проектом может работать команда.

Как не засорять операционную систему лишними программами? Как избежать конфликтов и несовместимости версий программного обеспечения? Как описать процесс настройки рабочего окружения так, чтобы другие члены команды смогли то же самое воспроизвести на своих компьютерах?

Vagrant создает изолированные среды разработки, основная система остается чистой. При этом не жертвуя привычными инструментами разработки (редакторы, браузеры и т. д.).

Описав процесс настройки в специальном файле, рабочее окружения легко пересоздать на другом компьютере. Vagrant работает в Windows, Mac и Linux.

Vagrant

Vagrant — это инструмент командной строки для создания виртуальных машин и управления ими.

По умолчанию Vagrant может подготавливать машины поверх VirtualBox, Hyper-V и Docker. Другие поставщики, такие как Libvirt (KVM), VMware и AWS, могут быть установлены через систему плагинов Vagrant.

Vagrant обычно используется разработчиками для создания среды разработки, которая работает в нескольких операционных системах. Vagrant помогает установить и настроить виртуальную машину на основе конфигурационного файла - **Vagrantfile**.

Vagrant работает с виртуальными машинами, а значит для виртуализации ему необходим какой-либо "**провайдер**". И для нашего примера подойдет **VirtualBox**.

Для Vagrant существуют готовые образы операционных систем, называемых «**коробками**»

» (**boxes**). **Коробка** — это, по сути, просто образ уже установленной и настроенной операционной системы, на основании которого строится виртуальная машина. Часто авторы коробок собирают их для поддержки в разных гипервизорах (приложениях для работы с виртуальными машинами).

Аналог Docker Hub только для образов ОС (коробок) - является **Vagrant Cloud**:
<https://app.vagrantup.com/boxes/search>.

Виртуальная машина, управляемая Vagrant-ом, представляет собой полноценный виртуальный сервер, на котором установлена операционная система и весь необходимый набор приложений и служб.

Чтобы развернуть коробку в виртуальную машину, нужен файл конфигурации, в котором описаны все параметры развертывания. Такой файл называется **Vagrantfile** и написан он с применением синтаксиса **Ruby** (на котором и написан Vagrant).

How Vagrant works

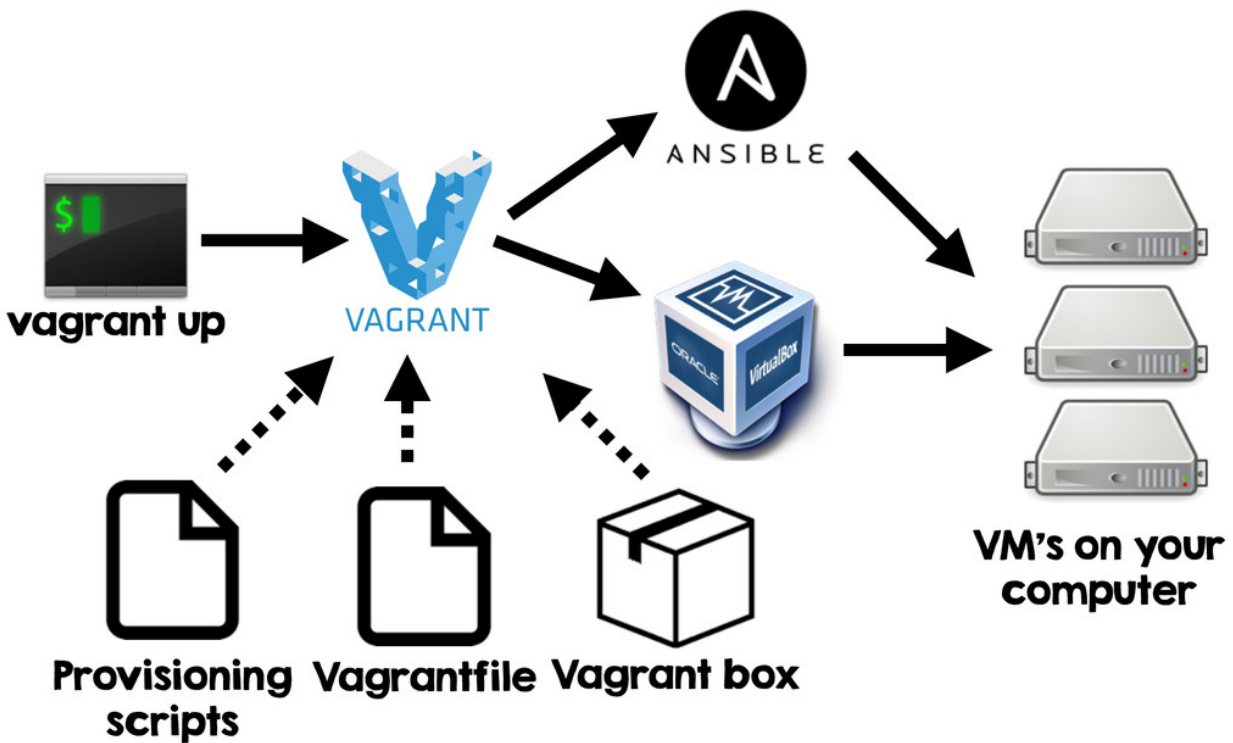
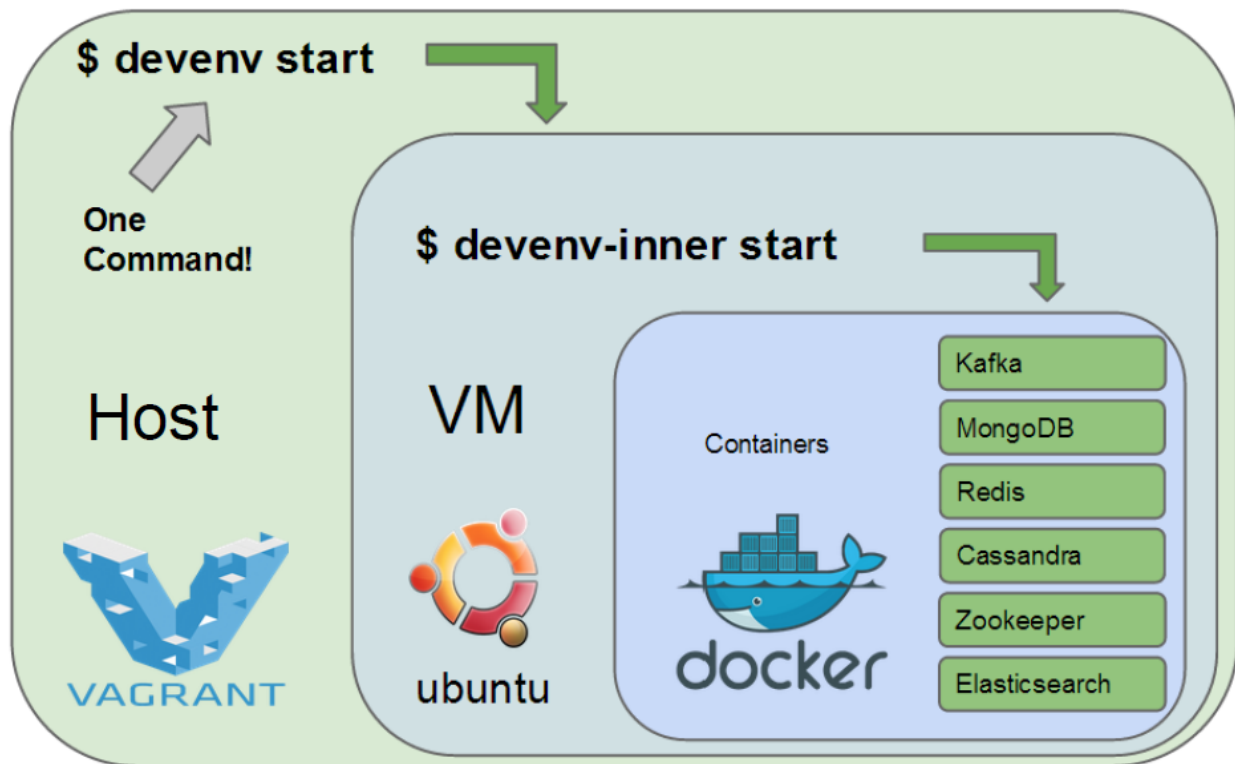


Схема работы Vagrant

Также ничего не мешает использовать Vagrant вместе с **Docker**.

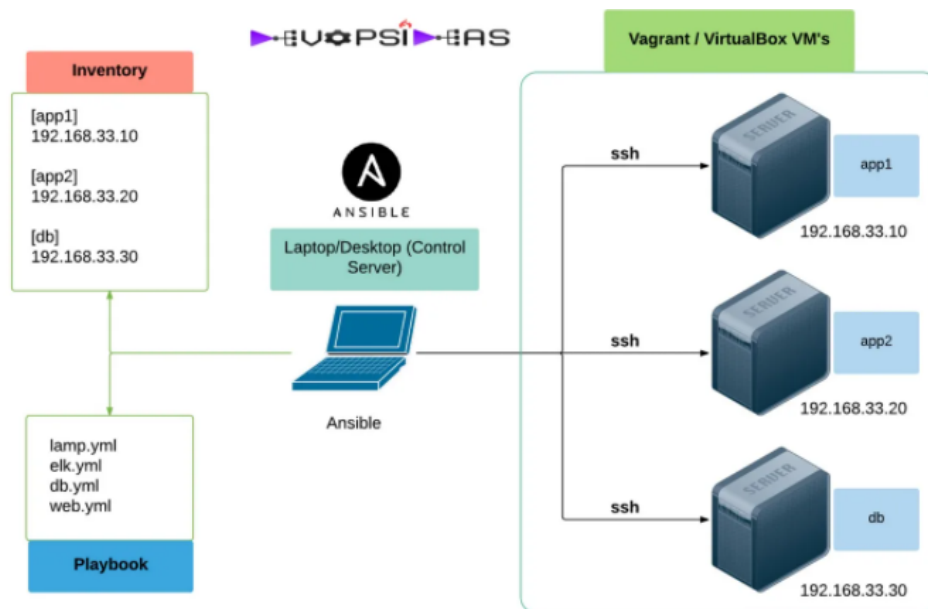
Мы можем использовать Vagrant — ради возможности создавать изолированную среду разработки, основанную на виртуальной машине со всеми необходимыми сервисами, которая должна будет воспроизвести реальную рабочую среду.

Другими словами, для быстрого и эффективного запуска Docker.



Использование Vagrant для создания виртуальной машины для docker-контейнеров

Кроме того, Vagrant часто применяют в связке с таким инструментом как [Ansible](#).



Ansible Local Testing with Vagrant and Virtualbox

Ansible использует Vagrant для запуска тестовых виртуальных машин

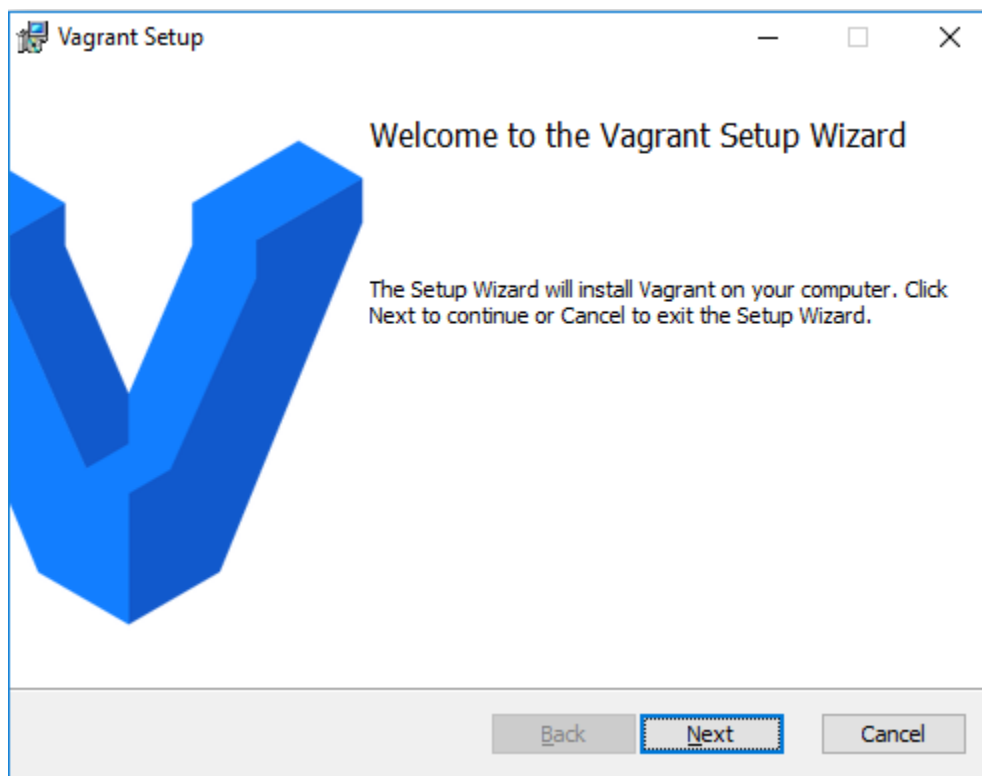
Таким образом, Vagrant нужен для удалённого и массового создания виртуальных машин, а Ansible для управления ими: настройки машин, установки и удаления на них ПО и т.д.

Установка Vagrant в Windows 10

Первоначально необходимо скачать и установить провайдера для виртуальных машин: VirtualBox, VMWare WS и т.д. Затем скачать и установить Vagrant.

Для установки Vagrant необходимо скачать пакет с файлами для установки, который доступен по ссылке: <https://releases.hashicorp.com/vagrant/>. В появившемся окне выбираем последнюю версию установщика. Для установки на Windows выбираем файл с расширением .msi.

После скачивания запускаем файл vagrant_x.x.x_x86_64.msi. В результате появится диалоговое окно с приветствием. Выполняем стандартную установку любого Windows-приложения (Далее-Далее-Далее).



Окно установщика Vagrant

После завершения установки Vagrant на Windows 10 появится окно с просьбой о перезагрузке компьютера.

После перезагрузки компьютера обязательно необходимо добавить переменную `vagrant` в системные переменные. Это нужно для того, чтобы в командной строке можно было вызывать Vagrant при помощи одной команды: `vagrant`.

Изменения в переменные среды мы будем вносить при помощи командной строки, а конкретно в переменную `PATH` мы будем добавлять путь до каталога, куда установлен Vagrant (в нашем случае это: `C:\HashiCorp\Vagrant\bin`).

Вызываем командную строку следующим образом:

1. нажимаем на клавиатуре клавиши `Ctrl+R` для вызова окна "Выполнить". 2. В появившемся окне набираем команду: `cmd`. 3. Нажимаем кнопку `OK`:

В окне командной строки добавляем следующую команду:

```
set PATH=%PATH%;путь_до_папки_c_vagrant\Vagrant\bin
```

В моём случае:

```
set PATH=%PATH%;D:\Vagrant\bin
```

Теперь для проверки того, удалось ли нам прописать путь к Vagrant в переменные среды, в этом же окне командной строки набираем команду:

```
vagrant -v
```

При помощи данной команды мы получим информацию о версии установленного Vagrant.

На этом процесс установки и настройки Vagrant и VirtualBox на Windows 10 завершен.

Установка Vagrant в Linux (Ubuntu 20.04)

Если [VirtualBox](#) не установлен в вашей системе, вы можете установить его, запуская:

```
sudo apt-get update  
sudo apt-get install virtualbox
```

Пакет Vagrant, доступный в репозиториях Ubuntu, не обновляется регулярно. Его следует скачивать с официального сайта Vagrant.

```
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
```

```
sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main"
```

```
sudo apt-get update && sudo apt-get install vagrant
```

Чтобы убедиться, что установка прошла успешно, выполните следующую команду, которая распечатает версию Vagrant:

```
vagrant --version
```

Если на экране появилась версия Vagrant, значит установка прошла успешно. Можно приступать к работе.

Создание виртуальной машины в Vagrant

Для создания виртуальной машины для начала создадим отдельную папку, например, *VagrantVM* и перейдём в неё:

```
mkdir VagrantVM  
cd VagrantVM
```

Наше web-приложение будет состоять из одного файла, которое будет выводить информацию о текущей конфигурации PHP (можно создать проект Python, Ruby, Golang и т.д.). Создайте в папке проекта файл `index.php` со следующим содержимым:

```
<?php  
  
phpinfo();
```

Теперь займемся созданием рабочего окружения в котором будет выполняться код приложения. Введите в терминале:

```
vagrant init -m bento/ubuntu-20.04
```

Команда `vagrant init` создает файл Vagrantfile в текущей директории. В нем на языке Ruby описывается конфигурация виртуальной машины. Синтаксис простой и понятный даже если вы никогда не использовали Ruby. Содержимое созданного файла Vagrantfile:


```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.vm.box = "bento/ubuntu-20.04"
end
```

При создании Vagrantfile мы указали название бокса `bento/ubuntu-20.04`. Бокс это образ операционной системы, который так же может содержать установленные программы (LAMP, Python и т.д).

Разработчики Vagrant рекомендуют использовать образы Bento, которые хранятся в специальном каталоге вместе с остальными образами.

Перед созданием виртуальной машины добавим еще несколько настроек в Vagrantfile:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  # Образ виртуальной машины с Vagrant Cloud
  config.vm.box = "bento/ubuntu-20.04"

  # Настройки виртуальной машины и выбор провайдера
  config.vm.provider "virtualbox" do |vb|
    vb.name = "VagrantVM"
    # Отключаем интерфейс, он не понадобится
    vb.gui = false
    # 2 Гб оперативной памяти
    vb.memory = "2048"
    # Одноядерный процессор
    vb.cpus = 1
  end

  config.vm.hostname = "VagrantVM"

  config.vm.synced_folder ".", "/home/vagrant/code",
    owner: "www-data", group: "www-data"
  # Переброс портов
  config.vm.network "forwarded_port", guest: 80, host: 8000
  config.vm.network "forwarded_port", guest: 3306, host: 33060
  # Команда для настройки сети
  config.vm.network "private_network", ip: "192.168.10.100"
  # Команда, которая выполнится после создания машины
  config.vm.provision "shell", path: "provision.sh"
end
```

- `config.vm.box` — базовый образ. В нашем случае Ubuntu 20.04
- `config.vm.provider` — система виртуализации. Доступные провайдеры: VirtualBox, VMware, Hyper-V, Docker и др.
- `config.vm.hostname` — имя хоста.
- `config.vm.synced_folder` — синхронизация папок. Первым параметром передается путь хостовой (основной) машины, вторым параметром передается путь гостевой (виртуальной машины)
- `config.vm.network` — настройки сети. Мы настроили перенаправление портов и статический IP-адрес, к которому привяжем домен. Доступны и другие настройки.
- `config.vm.provision` — служит для автоматической установки и настройки программного обеспечения. Мы использовали самый простой способ Shell-скрипт, также доступны другие: Ansible, Chef, Puppet и др.

Для нескольких виртуальных машин:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

hosts = {
  "conor" => "192.168.88.10",
  "mcgregor" => "192.168.88.11"
}

Vagrant.configure("2") do |config|
  config.vm.box = "bento/ubuntu-20.04"
  hosts.each do |name, ip|
    config.vm.define name do |machine|
      machine.vm.network :private_network, ip: ip
      machine.vm.provider "virtualbox" do |v|
        v.name = name
        v.gui = false
        v.memory = "1024"
        v.cpus = 1
      end
    end
    config.vm.provision "shell", path: "provision.sh"
    config.vm.network "forwarded_port", guest: 80, host: 8000
  end
end
```

Полный список опций для настройки представлен в официальной документации:
https://www.vagrantup.com/docs/vagrantfile/machine_settings

Для удобства настройка системы вынесена в отдельный файл `provision.sh`. Во время установки Vagrant запустит его внутри созданной виртуальной машины. Содержимое файла `provision.sh`:

```
apt-get update
apt-get -y upgrade

apt-add-repository ppa:ondrej/php -y

apt-get update

apt-get install -y software-properties-common curl zip

apt-get install -y php7.2-cli php7.2-fpm \
php7.2-pgsql php7.2-sqlite3 php7.2-gd \
php7.2-curl php7.2-memcached \
php7.2-imap php7.2-mysql php7.2-mbstring \
php7.2-xml php7.2-json php7.2-zip php7.2-bcmath php7.2-soap \
php7.2-intl php7.2-readline php7.2-ldap

apt-get install -y nginx

rm /etc/nginx/sites-enabled/default
rm /etc/nginx/sites-available/default

cat > /etc/nginx/sites-available/vagrantvm <<EOF
server {
    listen 80;
    server_name .vagrantvm.loc;
    root "/home/vagrant/code";

    index index.html index.htm index.php;

    charset utf-8;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location = /favicon.ico { access_log off; log_not_found off; }
    location = /robots.txt  { access_log off; log_not_found off; }

    access_log off;
    error_log /var/log/nginx/vagrantvm-error.log error;

    location ~ \.php$ {
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
```

```

        fastcgi_pass unix:/var/run/php/php7.2-fpm.sock;
        fastcgi_index index.php;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    }
}
EOF

ln -s /etc/nginx/sites-available/vagrantvm /etc/nginx/sites-enabled/vagrantvm

service nginx restart

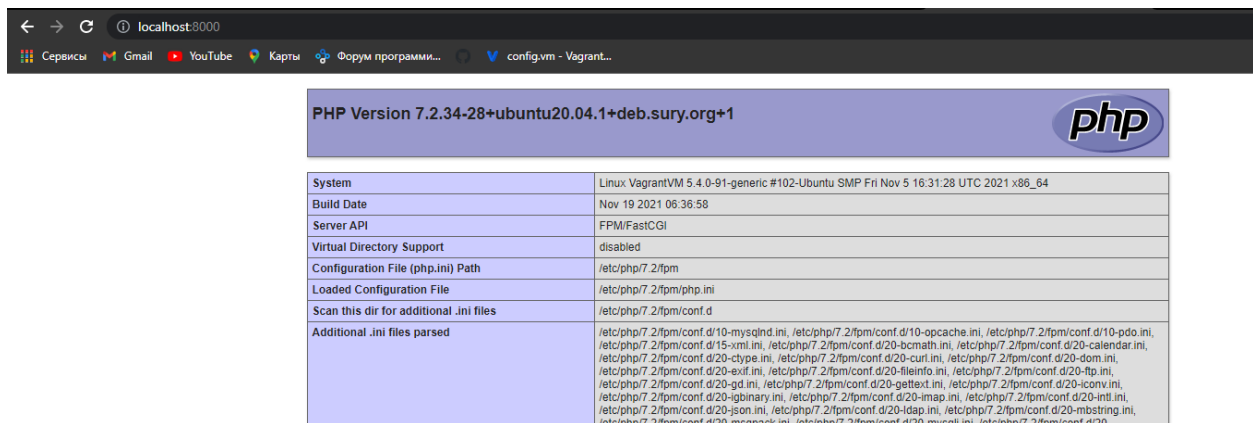
```

Создаем виртуальную машину. В первый раз процесс займет больше времени, Vagrant должен скачать образ с операционной системой. Выполните команду, которая создаст и запустит виртуальную машину:

```
vagrant up
```

Изолированное окружение готово! Откройте браузер и введите:

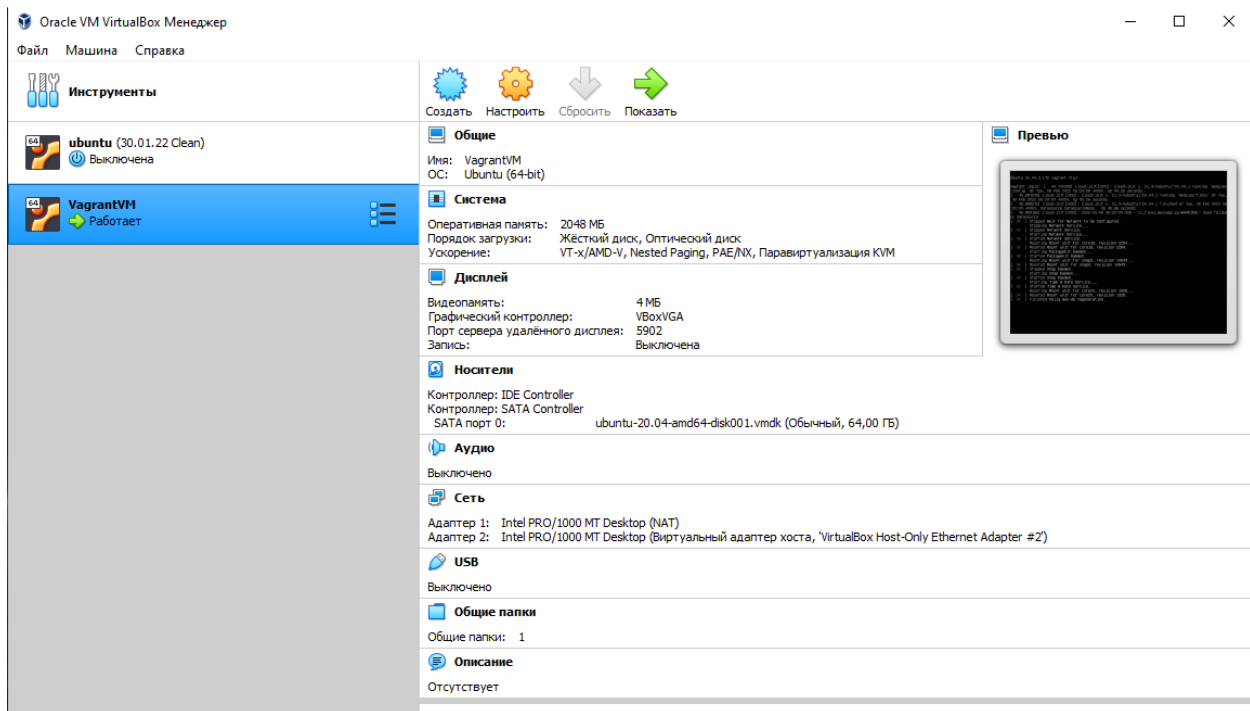
```
http://localhost:8000
```



PHP Version 7.2.34-28+ubuntu20.04.1+deb.sury.org+1

System	Linux VagrantVM 5.4.0-91-generic #102-Ubuntu SMP Fri Nov 5 16:31:28 UTC 2021 x86_64
Build Date	Nov 19 2021 06:36:58
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/fpm
Loaded Configuration File	/etc/php/7.2/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/fpm/conf.d
Additional .ini files parsed	/etc/php/7.2/fpm/conf.d/10-mysqlnd.ini, /etc/php/7.2/fpm/conf.d/10-opcache.ini, /etc/php/7.2/fpm/conf.d/10-pdo.ini, /etc/php/7.2/fpm/conf.d/15-xm1.ini, /etc/php/7.2/fpm/conf.d/20-bcmath.ini, /etc/php/7.2/fpm/conf.d/20-calendar.ini, /etc/php/7.2/fpm/conf.d/20-curl.ini, /etc/php/7.2/fpm/conf.d/20-dom.ini, /etc/php/7.2/fpm/conf.d/20-exif.ini, /etc/php/7.2/fpm/conf.d/20-fileinfo.ini, /etc/php/7.2/fpm/conf.d/20-ftp.ini, /etc/php/7.2/fpm/conf.d/20-gd.ini, /etc/php/7.2/fpm/conf.d/20-gettext.ini, /etc/php/7.2/fpm/conf.d/20-iconv.ini, /etc/php/7.2/fpm/conf.d/20-igbinary.ini, /etc/php/7.2/fpm/conf.d/20-imap.ini, /etc/php/7.2/fpm/conf.d/20-intl.ini, /etc/php/7.2/fpm/conf.d/20-json.ini, /etc/php/7.2/fpm/conf.d/20-ldap.ini, /etc/php/7.2/fpm/conf.d/20-mbstring.ini, /etc/php/7.2/fpm/conf.d/20-mysqli.ini, /etc/php/7.2/fpm/conf.d/20-mysqlnd.ini, /etc/php/7.2/fpm/conf.d/20-redis.ini, /etc/php/7.2/fpm/conf.d/20-sockets.ini, /etc/php/7.2/fpm/conf.d/20-sysvmsg.ini, /etc/php/7.2/fpm/conf.d/20-sysvsem.ini, /etc/php/7.2/fpm/conf.d/20-xml.ini, /etc/php/7.2/fpm/conf.d/20-xmlrpc.ini, /etc/php/7.2/fpm/conf.d/20-zip.ini, /etc/php/7.2/fpm/conf.d/20-zlib.ini

Результат работы



Отображение созданной виртуальной машины в VirtualBox

По умолчанию созданная виртуальная машина будет расположена в дефолтной папке VirtualBox для его виртуальных машин.

Основные команды Vagrant

Команды необходимо выполнять находясь внутри папки с виртуальной машиной

- `vagrant up` - запустить или создать виртуальную машину
- `vagrant reload` - перезагрузка виртуальной машины
- `vagrant halt` - останавливает виртуальную машину
- `vagrant destroy` - удаляет виртуальную машину
- `vagrant suspend` - "замораживает" виртуальную машину
- `vagrant global-status` - выводит список всех ранее созданных виртуальных машин в хост-системе
- `vagrant status` - посмотреть статус виртуальной машины

- `vagrant ssh` - подключается к виртуальной машине по SSH
- `vagrant` - список всех доступных команд

Подключение к виртуальной машине через SSH

Работать с этой виртуальной машиной можно через SSH. Для этого нам потребуется программа PuTTY, которую можно скачать [отсюда](#).

Так же можно с виртуальными машинами Vagrant работать и из командной строки. В том же окне командной строки выполняем команду:

```
vagrant ssh
```

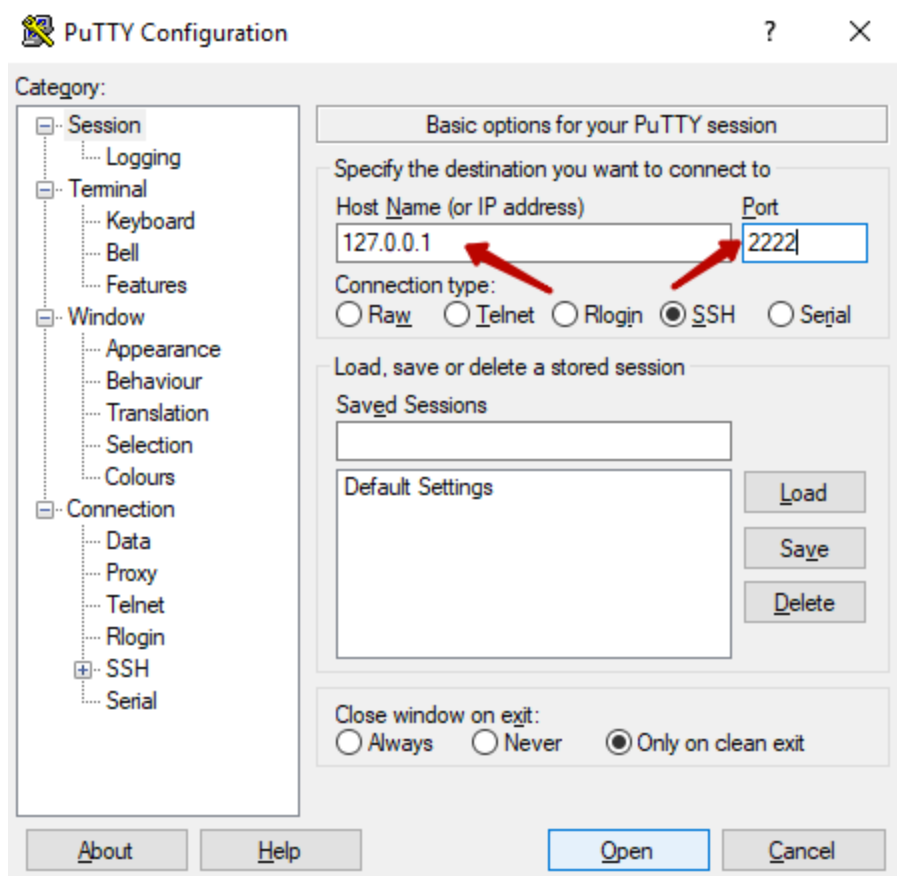
Пароль по умолчанию: *vagrant*. Имя пользователя: *vagrant*. Потому вводим здесь пароль: *vagrant*. И нажимаем клавишу Enter.

Здесь можно непосредственно работать с виртуальной машиной при помощи командной строки.

Чтобы вернуться из этого режима в нормальный режим командной строки хостовой ОС, выполняем команду:

```
exit
```

Для работы с виртуальной машиной через программу PuTTY, указываем следующие параметры для соединения с виртуальной машиной:



SSH подключение через Putty

Нажимаем кнопку **Open**. В появившемся диалоговом окне нажимаем **Да**. Далее необходимо ввести логин и пароль (как мы ранее определились, по умолчанию `vagrant/vagrant`).

```

D:\VagrantVM>vagrant ssh
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-91-generic x86_64)
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-91-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Wed 09 Feb 2022 05:26:29 AM UTC

System load:  0.0               Processes:            115
Usage of /:   12.5% of 30.88GB   Users logged in:     0
Memory usage: 15%              IPv4 address for eth0: 10.0.2.15
Swap usage:   0%               IPv4 address for eth1: 192.168.10.100

This system is built by the Bento project by Chef Software
More information can be found at https://github.com/chef/bento
vagrant@VagrantVM:~$ pwd
/home/vagrant
vagrant@VagrantVM:~$ ls -la
total 36
drwxr-xr-x 5 vagrant vagrant 4096 Feb  8 06:20 .
drwxr-xr-x 3 root    root    4096 Dec 19 19:42 ..
-rw-r--r-- 1 vagrant vagrant  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 vagrant vagrant 3771 Feb 25  2020 .bashrc
drwx----- 2 vagrant vagrant 4096 Dec 19 19:42 .cache
drwxrwxrwx 1 www-data www-data  0 Feb  8 06:15 code
-rw-r--r-- 1 vagrant vagrant  807 Feb 25  2020 .profile
drwx----- 2 vagrant root    4096 Feb  8 06:20 .ssh
-rw-r--r-- 1 vagrant vagrant    0 Dec 19 19:42 .sudo_as_admin_successful
-rw-r--r-- 1 vagrant vagrant    6 Dec 19 19:42 .vbox_version
-rw-r--r-- 1 root    root    180 Dec 19 19:44 .wget-hsts
vagrant@VagrantVM:~$

```

Результат подключения по SSH

Использование прокси для работы Vagrant

```

mkdir Vagrant
cd Vagrant/
vagrant init bento/ubuntu-20.04
export https_proxy='http://логин:пароль@ip_прокси:порт/'
export VAGRANT_HTTP_PROXY=${http_proxy}
export VAGRANT_NO_PROXY="127.0.0.1"
vagrant up

```

Заключение

Подводя итоги пройдемся по основным моментам рабочего процесса. Vagrant создает изолированные среды разработки, основная система остается чистой. При этом не жертвуя привычными инструментами разработки (редакторы, браузеры и т. д.). В корне проекта вместе с исходным кодом лежит файл Vagrantfile. Для разворачивания рабочего окружения достаточно выполнить одну команду `vagrant up`. Если Vagrantfile изменился, выполните команды `vagrant destroy` и `vagrant up`, виртуальная машина будет пересоздана с новыми настройками. Используя систему контроля версий члены команды также легко пересоздадут окружение на своих компьютерах.

Каждый день, работая над проектом, запускаем сначала виртуальную машину командой `vagrant up`. Исходный код проекта автоматически синхронизируется с виртуальной машиной. Порты проброшены. Работаем над проектом в привычном редакторе кода и браузере. Если по каким-то причинам необходимо попасть в виртуальную машину, выполняем команду `vagrant ssh`. Закончив работу останавливаем виртуальную машину командой `vagrant halt`.

Список использованных источников

1. <https://evgeniyprokopev.com/vagrant-samyj-prostoj-i-bystryj-sposob-sozdaniya-izolirovannoj-sredy-razrabotki/>
2. <https://site2go.ru/article/ustanovka-i-nastrojka-vagrant-na-windows-10>
3. <https://routerus.com/how-to-install-vagrant-on-ubuntu-20-04/>
4. <https://robotuts.github.io/getting-started/vagrant/>
5. <https://www.vagrantup.com/docs>
6. <https://app.vagrantup.com/boxes/search>