vag
OTUS

Методическое пособие по выполнению домашнего задания по курсу «Администратор Linux. Professional»

Vagrant-стенд для обновления ядра и создания образа системы

Цель домашнего задания

Научиться обновлять ядро в ОС Linux. Получение навыков работы с Vagrant, Packer и публикацией готовых образов в Vagrant Cloud.

Описание домашнего задания

1) Обновить ядро ОС из репозитория ELRepo
2) Создать Vagrant box с помощью Packer
3) Загрузить Vagrant box в Vagrant Cloud

Дополнительные задания:

- Ядро собрано из исходников
- В образе нормально работают VirtualBox Shared Folders

Введение

Данные методические рекомендации помогут разобраться с инструментами Vagrant и Packer, а также получить базовые навыки работы с системами контроля версий. Эти технологии помогут Вам в выполнении следующих домашних работ.

Рассмотрим подробнее данные инструменты:

- **Vagrant** — ПО для создания и конфигурирования виртуальной среды.
- **Packer** — ПО для создания образов виртуальных машин.
- **Git** — система контроля версий
- **GitHub** — веб-сервис для хостинга IT-проектов и их совместной разработки.

В современном мире операционные системы постоянно обновляются, при обновлении ОС закрываются уязвимости программного обеспечения и исправляются ошибки. Обновление операционной системы является одним из важных навыков администратора Linux.

Другой важной задачей администратора является создание образов виртуальных машин. Это процесс значительно может сэкономить время специалиста. Представьте себе ситуацию: Вы развернули виртуальную машину, установили ОС, выполнили настройку. Выполнение всех этих действий может занять несколько часов. Если таких виртуальных машин потребуется несколько, то каждый раз придётся вручную создавать ВМ, устанавливать и настраивать её. Образы виртуальных машин помогут избежать ошибок во время развертывания и значительно сэкономить время администратору.

В данном руководстве будет рассмотрен процесс обновления ядра Linux в ОС, а также дальнейшее создание образа данной виртуальной машины и её загрузка в Vagrant Cloud.

Функциональные и нефункциональные требования

- ПК на Unix с 8ГБ ОЗУ или виртуальная машина с включенной Nested Virtualization.
- Созданный аккаунт на GitHub - https://github.com/
- Созданный аккаунт в Vagrant Cloud - https://app.vagrantup.com/boxes/search
- *Если Вы находитесь в России, для корректной работы Вам может потребоваться VPN.*

Предварительно установленное и настроенное следующее ПО:

- Hashicorp Vagrant (https://www.vagrantup.com/downloads)
- Oracle VirtualBox (https://www.virtualbox.org/wiki/Linux_Downloads).
- Hashicorp Packer (https://www.packer.io/downloads)
- Любой редактор кода, например Visual Studio Code, Atom и т.д.

Инструкция по выполнению домашнего задания

Все дальнейшие действия были проверены при использовании Vagrant 2.2.19, VirtualBox v6.1.32 и Packer v1.8.1. В лабораторной работе используются Vagrant boxes с CentOS 8 Stream (версия 20210210.0). Серьёзные отступления от этой конфигурации могут потребовать адаптации с вашей стороны.

Создадим Vagrantfile, в котором будут указаны параметры нашей ВМ:

```
# Описываем Виртуальные машины
MACHINES = {
  # Указываем имя ВМ "kernel update"
  :"kernel-update" => {
              #Какой vm box будем использовать
              :box_name => "centos/stream8",
              #Указываем box_version
              :box_version => "20210210.0",
              #Указываем количество ядер ВМ
              :cpus => 2,
              #Указываем количество ОЗУ в мегабайтах
              :memory => 1024,
            }
}

Vagrant.configure("2") do |config|
  MACHINES.each do |boxname, boxconfig|
    # Отключаем проброс общей папки в ВМ
    config.vm.synced_folder ".", "/vagrant", disabled: true
    # Применяем конфигурацию ВМ
    config.vm.define boxname do |box|
```

```
      box.vm.box = boxconfig[:box_name]
      box.vm.box_version = boxconfig[:box_version]
      box.vm.host_name = boxname.to_s
      box.vm.provider "virtualbox" do |v|
        v.memory = boxconfig[:memory]
        v.cpus = boxconfig[:cpus]

      end
    end
  end
end
```

После создания Vagrantfile, запустим виртуальную машину командой *vagrant up*. Будет создана виртуальная машина с ОС CentOS 8 Stream, с 2-мя ядрами CPU и 1ГБ ОЗУ.

Обновление ядра

Подключаемся по ssh к созданной виртуальной машины. Для этого в каталоге с нашим Vagrantfile вводим команду *vagrant ssh*

Перед работами проверим текущую версию ядра:

```
[vagrant@kernel-update ~]$ uname -r
4.18.0-277.el8.x86_64
```

Далее подключим репозиторий, откуда возьмём необходимую версию ядра:
sudo yum install -y
https://www.elrepo.org/elrepo-release-8.el8.elrepo.noarch.rpm

В репозитории есть две версии ядер:

- kernel-ml — свежие и стабильные ядра
- kernel-lt — стабильные ядра с длительной версией поддержки, более старые, чем версия ml.

Установим последнее ядро из репозитория elrepo-kernel:
sudo yum --enablerepo elrepo-kernel install kernel-ml -y

Параметр --enablerepo elrepo-kernel указывает что пакет ядра будет запрошен из репозитория elrepo-kernel.

Уже на этом этапе можно перезагрузить нашу виртуальную машину и выбрать новое ядро при загрузке ОС.

Если требуется, можно назначить новое ядро по-умолчанию вручную:
1) Обновить конфигурацию загрузчика:
sudo grub2-mkconfig -o /boot/grub2/grub.cfg
2) Выбрать загрузку нового ядра по-умолчанию:
sudo grub2-set-default 0

Далее перезагружаем нашу виртуальную машину с помощью команды sudo reboot

После перезагрузки снова проверяем версию ядра (версия должа стать новее):

```
[vagrant@kernel-update ~]$ uname -r
5.18.3-1.el8.elrepo.x86_64
```

На этом обновление ядра закончено

Создание образа системы

Для создания образа системы, нам потребуется программа Packer

Давайте кратко разберем, как Packer будет создавать образ нашей виртуальной машины:

- Сначала скачивается ISO-образ и сверяется его контрольная сумма, если контрольная сумма не совпадает, процесс создания образа завершается с ошибкой
- Далее идёт процесс установки ОС, его также можно представить в виде файла и Packer сам развернёт нам виртуальную машину, опираясь на инструкцию из этого файла
- После развертывания ВМ мы запустим скрипты настройки
- После выполнения всех вышеуказанных действий packer снимает образ
  В следующий раз мы можем просто загрузить данный образ и наша ВМ сразу развернётся без остальных настроек.

В каталоге с нашим Vagrantfile создадм папку packer: *mkdir packer*
Перейдём в эту папку: *cd packer*

Основным файлом для packer json-файл. В нём описываются все шаги создания образа. С помощью любого редактора кода создадим файл centos.json в каталоге packer и добавим в него следующее содержимое:

```
#Основная секция, в ней указываются характеристики нашей ВМ
{"builders": [
    {
      #Указываем ссылку на файл автоматической конфигурации
      "boot_command": [
        "<tab> inst.text inst.ks=http://{{ .HTTPIP }}:{{ .HTTPPort
}}/ks.cfg<enter><wait>"
      ],
      "boot_wait": "10s",
      #Указываем размер диска для ВМ
      "disk_size": "10240",
      "export_opts": [
        "--manifest",
        "--vsys",
        "0",
        "--description",
        "{{user `artifact_description`}}",
        "--version",
        "{{user `artifact_version`}}"
      ],
      #Указываем семейство ОС нашей ВМ
      "guest_os_type": "RedHat_64",
      #Указываем каталог, из которого возьмём файл автоматической конфигурации
      "http_directory": "http",
      #Контрольная сумма ISO-файла
      #Проверяется после скачивания файла
```

```
      "iso_checksum":
"b4bb35e2c074b4b9710419a9baa4283ce4a02f27d5b81bb8a714b576e5c2df7a",
      #Ссылка на дистрибутив из которого будет разворачиваться наша ВМ
      "iso_url":
"http://mirror.linux-ia64.org/centos/8-stream/isos/x86_64/CentOS
-Stream-8-x86_64-20230209-boot.iso",
      #Hostname нашей ВМ
      #Имя ВМ будет взято из переменной image_name
      "name": "{{user `image_name`}}",
      "output_directory": "builds",
      "shutdown_command": "sudo -S /sbin/halt -h -p",
      "shutdown_timeout": "5m",
      #Пароль пользователя
      "ssh_password": "vagrant",
      #Номер ssh-порта
      "ssh_port": 22,
      "ssh_pty": true,
      #Тайм-аут подключения по SSH
      #Если через 20 минут не получается подключиться, то сборка отменяется
      "ssh_timeout": "20m",
      #Имя пользователя
      "ssh_username": "vagrant",
      #Тип созданного образа (Для VirtualBox)
      "type": "virtualbox-iso",
      #Параметры ВМ
      #2 CPU и 1Гб ОЗУ
      "vboxmanage": [
        [
          "modifyvm",
          "{{.Name}}",
          "--memory",
          "1024"
        ],
        [
          "modifyvm",
          "{{.Name}}",
          "--cpus",
          "2"
        ]
      ],
      #Имя ВМ в VirtualBox
      "vm_name": "packer-centos-vm"
    }
  ],
  "post-processors": [
    {
      #Уровень сжатия
      "compression_level": "7",
      #Указание пути для сохранения образа
      #Будет сохранён в каталог packer
      "output":
"centos-{{user`artifact_version`}}-kernel-5-x86_64-Minimal.box",
      "type": "vagrant"
    }
  ],
  #Настройка ВМ после установки
```

```
  "provisioners": [
    {
      "execute_command": "{{.Vars}} sudo -S -E bash '{{.Path}}'",
      "expect_disconnect": true,
      "override": {
        "{{user `image_name`}}": {
          #Скрипты, которые будут запущены после установки ОС
          #Скрипты выполняются в указанном порядке
          "scripts": [
            "scripts/stage-1-kernel-update.sh",
            "scripts/stage-2-clean.sh"
          ]
        }
      },
      #Тайм-аут запуска скриптов, после того, как подключились по SSH
      "pause_before": "20s",
      "start_retry_timeout": "1m",
      "type": "shell"
    }
  ],
  #Указываем переменные
  #К переменным можно обращаться внутри данного JSON-файла
  "variables": {
    "artifact_description": "CentOS Stream 8 with kernel 5.x",
    "artifact_version": "8",
    "image_name": "centos-8"
  }
}
```

Информацию об остальных параметрах можно посмотреть в документации Packer.

Важно! При создании вашего JSON-файла не используйте комментарии (зеленый текст, который начинается со знака #)

Изучив JSON-файл, мы видим в нём ссылки на другие файлы:

В разделе "boot_command": указывается файл **ks.cfg** — это файл автоматической конфигурации ОС. Если говорить простыми словами, то этот файл установит ОС автоматически: укажет раскладку клавиатуры, часовой пояс, откуда скачать недостающие образы и т. д.

Далее в файле, мы видим, что файл **ks.cfg** берётся из каталога **http: f**

```
#Указываем каталог, из которого возьмём файл автоматической конфигурации
      "http_directory": "http",
```

Создадим в каталоге packer каталог http: *mkdir http*
Перейдём в каталог http: *cd http*

Создадим внутри каталога http файл **ks.cfg** со следующим содержимым:

```
# Подтверждаем лицензионное соглашение
eula --agreed

# Указываем язык нашей ОС
```

```
lang en_US.UTF-8
# Раскладка клавиатуры
keyboard us
# Указываем часовой пояс
timezone UTC+3

# Включаем сетевой интерфейс и получаем ip-адрес по DHCP
network --bootproto=dhcp --device=link --activate
# Задаём hostname otus-c8
network --hostname=otus-c8

# Указываем пароль root пользователя
rootpw vagrant
authconfig --enableshadow --passalgo=sha512
# Создаём пользователя vagrant, добавляем его в группу Wheel
user --groups=wheel --name=vagrant --password=vagrant --gecos="vagrant"

# Включаем SELinux в режиме enforcing
selinux --enforcing
# Выключаем штатный межсетевой экран
firewall --disabled

firstboot --disable

# Выбираем установку в режиме командной строки
text
# Указываем адрес, с которого установщик возьмёт недостающие компоненты
url --url="http://mirror.centos.org/centos/8-stream/BaseOS/x86_64/os/"

# System bootloader configuration
bootloader --location=mbr --append="ipv6.disable=1 crashkernel=auto"

skipx
logging --level=info
zerombr
clearpart --all --initlabel
# Автоматически размечаем диск, создаём LVM
autopart --type=lvm
# Перезагрузка после установки
reboot
```

Более подробно изучить остальные параметры можно в документации RedHat по автоматическому конфигурированию (Kickstart commands and options reference)

В разделе "provisioners": мы видим 2 скрипта, которые должны будут выполниться после установки ОС:

```
"scripts": [
            "scripts/stage-1-kernel-update.sh",
            "scripts/stage-2-clean.sh"
          ]
```

Для удобства скрипты выведены в отдельный каталог.

Вернёмся в каталог packer из каталога http: *cd ..*
Создадим в каталоге packer каталог scripts: *mkdir scripts*
Перейдём в каталог scripts: *cd scripts*

В каталоге scripts создадим 2 файла:

*В первом файле **stage-1-kernel-update.sh** будут содаржаться команды по обновлению ядра, которые мы разобрали в предыдущей части:*

```
#!/bin/bash

# Установка репозитория elrepo
sudo yum install -y
https://www.elrepo.org/elrepo-release-8.el8.elrepo.noarch.rpm
# Установка нового ядра из репозитория elrepo-kernel
yum --enablerepo elrepo-kernel install kernel-ml -y

# Обновление параметров GRUB
grub2-mkconfig -o /boot/grub2/grub.cfg
grub2-set-default 0
echo "Grub update done."
# Перезагрузка ВМ
shutdown -r now
```

*Второй файл **stage-2-clean.sh** просто очистит ненужные файлы из нашей ОС и добавит ssh-ключ пользователя vagrant (в данной практической работе его можно просто скопировать, разбирать его мы не будем):*

```
#!/bin/bash

# Обновление и очистка всех ненужных пакетов
yum update -y
yum clean all


# Добавление ssh-ключа для пользователя vagrant
mkdir -pm 700 /home/vagrant/.ssh
curl -sL
https://raw.githubusercontent.com/mitchellh/vagrant/master/keys/vagrant.pub
-o /home/vagrant/.ssh/authorized_keys
chmod 0600 /home/vagrant/.ssh/authorized_keys
chown -R vagrant:vagrant /home/vagrant/.ssh


# Удаление временных файлов
rm -rf /tmp/*
rm  -f /var/log/wtmp /var/log/btmp
rm -rf /var/cache/* /usr/share/doc/*
rm -rf /var/cache/yum
rm -rf /vagrant/home/*.iso
rm  -f ~/.bash_history
history -c
```

```
rm -rf /run/log/journal/*
sync
grub2-set-default 0
echo "###   Hi from second stage" >> /boot/grub2/grub.cfg
```

После того, как мы создали все необходимые файлы, мы можем приступить к сборке нашего образа.

Переходим в каталог packer: *cd ..*
Создадим образ системы: *packer build centos.json*

Далее начнётся процесс по созданию образа системы. Процесс может занять продолжительное время. (более 10 минут). Packer будет показывать этапы создания образа:

```
→  packer packer build centos.json
centos-8: output will be in this color.

==> centos-8: Retrieving Guest additions
==> centos-8: Trying /usr/share/virtualbox/VBoxGuestAdditions.iso
==> centos-8: Trying /usr/share/virtualbox/VBoxGuestAdditions.iso
==> centos-8: /usr/share/virtualbox/VBoxGuestAdditions.iso => /usr/share/virtualbox/VBoxGuestAdditio
ns.iso
==> centos-8: Retrieving ISO
==> centos-8: Trying http://mirror.linux-ia64.org/centos/8-stream/isos/x86_64/CentOS-Stream-8-x86_64
-20220603-boot.iso
==> centos-8: Trying http://mirror.linux-ia64.org/centos/8-stream/isos/x86_64/CentOS-Stream-8-x86_64
-20220603-boot.iso?checksum=sha256%3A0f433162c0ba7b845b06a9cb3c64768b88da1ddf9fb921b65de2a055fb7453f
7
==> centos-8: http://mirror.linux-ia64.org/centos/8-stream/isos/x86_64/CentOS-Stream-8-x86_64-202206
03-boot.iso?checksum=sha256%3A0f433162c0ba7b845b06a9cb3c64768b88da1ddf9fb921b65de2a055fb7453f7 => /h
ome/alex/.cache/packer/8139869fd07dc0a66aa70fed551459687d9ac9c0.iso
==> centos-8: Starting HTTP server on port 8419
==> centos-8: Creating virtual machine...
==> centos-8: Creating hard drive builds/packer-centos-vm.vdi with size 10240 MiB...
==> centos-8: Mounting ISOs...
    centos-8: Mounting boot ISO...
==> centos-8: Creating forwarded port mapping for communicator (SSH, WinRM, etc) (host port 3410)
```

После успешного создания образа в Packer выдаст следующее сообщение:
==> Builds finished. The artifacts of successful builds are:
--> centos-8: 'virtualbox' provider box:
centos-8-kernel-5-x86_64-Minimal.box

Также после успешного создания образа в каталоге packer должен появиться файл centos-8-kernel-5-x86_64-Minimal.box

Если вдруг процесс завершится неудачно, посмотрите, на ошибки, которые были в ходе создания образа. По ним будет понятно, где была допущена ошибка. После исправления ошибки можно снова запустить сборку конмадой packer build centos.json

После создания образа, его рекомендуется проверить. Для проверки импортируем полученный vagrant box в Vagrant:
vagrant box add centos8-kernel5 centos-8-kernel-5-x86_64-Minimal.box

Проверим, что образ теперь есть в списке имеющихся образов vagrant:
```
➜  packer vagrant box list
centos/7          (virtualbox, 2004.01)
```

```
centos/stream8     (virtualbox, 20210210.0)
centos8-kernel5    (virtualbox, 0)
debian/bullseye64  (virtualbox, 11.20211230.1)
rockylinux/8       (virtualbox, 4.0.0)
ubuntu/bionic64    (virtualbox, 20220131.0.0)
ubuntu/focal64     (virtualbox, 20211026.0.0)
➜  packer
```

Создадим Vagrantfile на основе образа centos8-kernel5:
vagrant init centos8-kernel5

В каталоге packer появится Vagrantfile

Запустим нашу ВМ: *vagrant up*
Подключимся к ней по SSH: *vagrant ssh*
Проверим версию ядра:
[vagrant@otus-c8 ~]$ ***uname -r***
5.18.5-1.el8.elrepo.x86_64
[vagrant@otus-c8 ~]$

Как мы видим в нашей ВМ сразу используется новая версия ядра.

Далее можно выйти из нашей ВМ: ***exit***
И удалить ненужную виртуальную машину: ***vagrant destory --force***

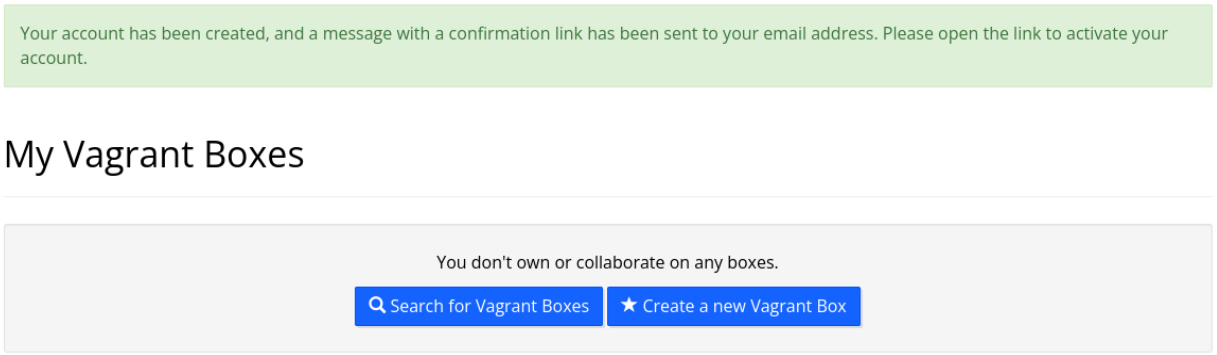
На этом процесс создания и проверки образа завершен.

Загрузка образа в Vagrant cloud

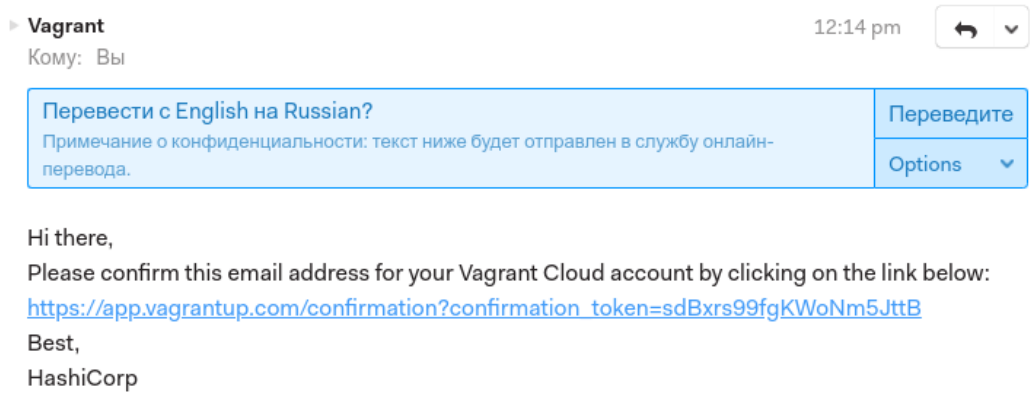
Для того, чтобы загрузить наш образ в Vagrant Cloud нам потребуется учётная запись.

Для этого заходим на веб-станицу https://app.vagrantup.com/ и выбираем пункт **Create an Account** *(Если у Вас уже есть аккаунт в Vagrant Cloud, то пропустите этот пункт). Далее заполняем поля и нажимаем* ***Create acount.***

После удачной регистрации переда Вами откроется ваш аккаунт:



Далее Вам на указанную почту должно прийти письмо с просьбой подтвердить регистрацию.



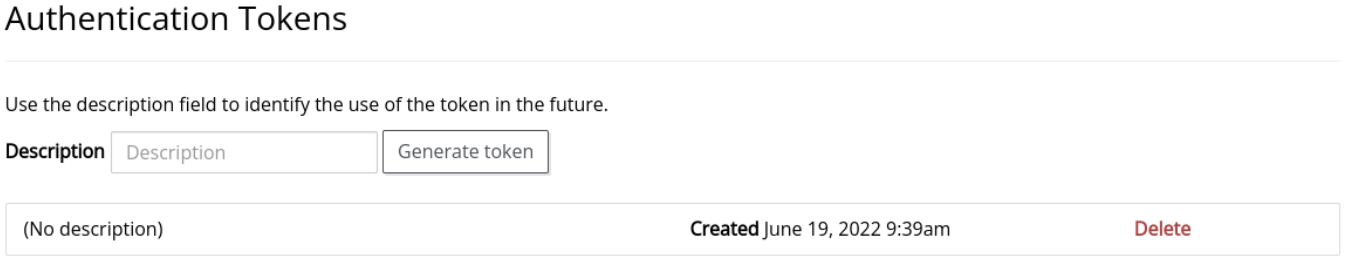
Перейдём по ссылке и подтвердим указанный e-mail.

Далее, находясь в учётной записи Vagrant cloud **щёлкаём по вашему логину** в правой верхней части экрана и заходим в раздел **Settings**.

Откроется страница настроек

переходим в раздел **Security** и в разделе **Authentication Tokens** нажимаем **Generate token**

После этого Вам будет создан токен, он отобразится только 1 раз, скопируйте его себе и сохраните. ***Далее в настройках нельзя будет посмотреть ваш токен.***



Теперь добавим в Vagrant Cloud наш vagrant box:
Если Вы находитесь в России, то на этом этапе требуется включить VPN. *(Настройка и запуск VPN были разобраны в методичке «Настройка рабочего места»)*

Находясь в каталоге **packer** логинимся в vagrant cloud: *vagrant cloud auth login*
Далее потребуется последовательно указать:

- Ваш логин
- Ваш пароль
- Ваш созданный токен
  Если все реквизиты доступа будут указанны правильно, вы увидите сообщение: **You are now logged in.**

  Далее публикуем наш образ:
  vagrant cloud publish --release <user_account>/centos8-kernel5 1.0 virtualbox centos-8-kernel-5-x86_64-Minimal.box

  Разберем подрбнее данную команду:
- *cloud publish* - загрузить образ в облако;
- *release* - указывает на необходимость публикации образа после загрузки;
- *<username>/centos8-kernel5* - username, указаный при публикации и имя образа;
- *1.0* - версия образа;
- *virtualbox* - провайдер;
- *centos-8-kernel-5-x86_64-Minimal.box* - имя файла загружаемого образа;

  Процесс загрузки образа может занять продолжительное время. После успешной загрузки Мы получим сообщение: *Complete! Published <username>/centos8-kernel5*
  Далее будут перечисленны метаданные образа.

Критерии оценивания

Статус «Принято» ставится при выполнении следующих условий:

1. Ссылка на репозиторий GitHub в котором находятся файлы указанные в последующих пунктах.
2. Vagrantfile, который будет разворачивать виртуальную машину используя vagrant box который вы загрузили Vagrant Cloud.
3. Документация по каждому заданию:
Создайте файл README.md и снабдите его следующей информацией:
- название выполняемого задания;
- текст задания;
- описание команд и их вывод;
- особенности проектирования и реализации решения,
- заметки, если считаете, что имеет смысл их зафиксировать в репозитории.

Рекомендуемые источники

- Репозиторий manual_kernel_update - https://github.com/dmitry-lyutenko/manual_kernel_update/blob/master/manual/manual.md
- Статья о GitHub - https://ru.wikipedia.org/wiki/GitHub
- Elrepo HomePage - http://elrepo.org/tiki/HomePage
- Packer Docs - https://www.packer.io/docs
- Параметры автоконфигурации от RedHat - https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/performing_an_advanced_rhel_installation/kickstart-commands-and-options-reference_installing-rhel-as-an-experienced-user