Отчёт 10 лаба:

$ export GITHUB\_USERNAME=<имя\_пользователя>

$ export PACKAGE\_MANAGER=<пакетный\_менеджер>

$ cd ${GITHUB\_USERNAME}/workspace # переходим в папку проекта

$ ${PACKAGE\_MANAGER} install vagrant # устанавливаем vagrant

$ vagrant version # проверяем версию

# создаем рабочее окружение в котором будет выполняться код,Команда vagrant

init создает файл Vagrantfile в текущей директории.

$ vagrant init bento/ubuntu-19.10

$ less Vagrantfile # просматриваем содержимое файла

# флаг -m позволяет создать минимальный Vagrantfile

# флаг -f создает файл Vagrant, перезаписав его по текущему пути:

$ vagrant init -f -m bento/ubuntu-19.10

$ mkdir shared # создвем директорию

$ cat > Vagrantfile <<EOF

\$script = <<-SCRIPT

# команда установки докера

sudo apt install docker.io -y

# выгрузка образа

sudo docker pull fastide/ubuntu:19.04

# создание контейнера

sudo docker create -ti --name fastide fastide/ubuntu:19.04 bash

# копируем содержимое из файловой системы контейнера в локальную машину

sudo docker cp fastide:/home/developer /home/

# регистрация пользователя

sudo useradd developer

# -G - дополнительные группы для пользователя;

# -a - добавить пользователя в дополнительные группы из параметра -G, а не заменять им текущее значение;

sudo usermod -aG sudo developer

# устанавливаем пароль

echo "developer:developer" | sudo chpasswd

# даем доступ новому пользователю

sudo chown -R developer /home/developer

SCRIPT

EOF

$ cat >> Vagrantfile <<EOF

# 2 - версия Vagrant, do определяет начало файла

Vagrant.configure("2") do |config|

# оперделение плагина для локального проекта

config.vagrant.plugins = ["vagrant-vbguest"]

EOF

$ cat >> Vagrantfile <<EOF

# указываем бокс

config.vm.box = "bento/ubuntu-19.10"

# создаем частную сеть

config.vm.network "public\_network"

# настраиваем синхронизированные папки на компьютере, чтобы папки на хост-компьютере могли синхронизироваться с гостевым компьютером и с него.

config.vm.synced\_folder('shared', '/vagrant', type: 'rsync')

# настрока провайдера

config.vm.provider "virtualbox" do |vb|

# запуск Vagrant машины в графичнском режиме

vb.gui = true

# выделяем оперативную память

vb.memory = "2048"

end

# добавляем секцию для провизионинга виртуалки shell-скриптом:

config.vm.provision "shell", inline: \$script, privileged: true

# передаем команду в исполняемый файл

config.ssh.extra\_args = "-tt"

end

EOF

# проверка файла Vagrantfile

$ vagrant validate

# проверка состояния виртуальной машины

$ vagrant status

# эта команда создает и настраивает гостевые машины в соответствии с файлом Vagrantfile.

$ vagrant up # --provider virtualbox

# Команда port отображает полный список гостевых портов, сопоставленных с портами хост-машины

$ vagrant port

$ vagrant status

# подключение к машине

$ vagrant ssh

# Эта команда используется для управления моментальными снимками на гостевой машине. Моментальные снимки записывают состояние гостевой машины в данный момент времени

$ vagrant snapshot list

# Это делает снимок и помещает его в стек снимков.

$ vagrant snapshot push

$ vagrant snapshot list

# Эта команда отключает работающую машину, которой управляет Vagrant.

$ vagrant halt

#Эта команда является обратнойvagrant snapshot push: она восстановит нажатое состояние.

$ vagrant snapshot pop

config.vm.provider :vmware\_esxi do |esxi|

# задаем имя хоста и тд

esxi.esxi\_hostname = '<exsi\_hostname>'

esxi.esxi\_username = 'root'

esxi.esxi\_password = 'prompt:'

esxi.esxi\_hostport = 22

esxi.guest\_name = '${GITHUB\_USERNAME}'

esxi.guest\_username = 'vagrant'

esxi.guest\_memsize = '2048'

esxi.guest\_numvcpus = '2'

esxi.guest\_disk\_type = 'thin'

end

# устанавливаем плагин

$ vagrant plugin install vagrant-vmware-esxi

# просматриваем плагины

$ vagrant plugin list

$ vagrant up --provider=vmware\_esxi