Сборник практических работ для курса «AL-1724VR. Установка и управление виртуализацией в ОС Astra Linux Special Edition 1.7»

Требования:

- Один компьютер для каждого слушателя и преподавателя в составе:
  - операционная система: ОС Astra Linux Special Edition версии 1.7.4 или новее;
  - ядро ОС 5.15.0-70-generic или новее;
  - процессор архитектуры: x86-64 не менее 4 ядер;
  - оперативная память: не менее 16 Гб;
  - дисковая подсистема: NVME или SATA SSD, не менее 128 Гб свободного пространства.
- Программное обеспечение:
  - графический интерфейс Fly;
  - средства работы с Интернет (браузер Firefox);
  - консольные утилиты (терминал Fly).
- Доступ компьютеров в сеть Интернет.
- Административная учетная запись пользователя (входит в группу astra-admin)

Материалы:

- Виртуальный диск **AL-1742V-template.qcow2**

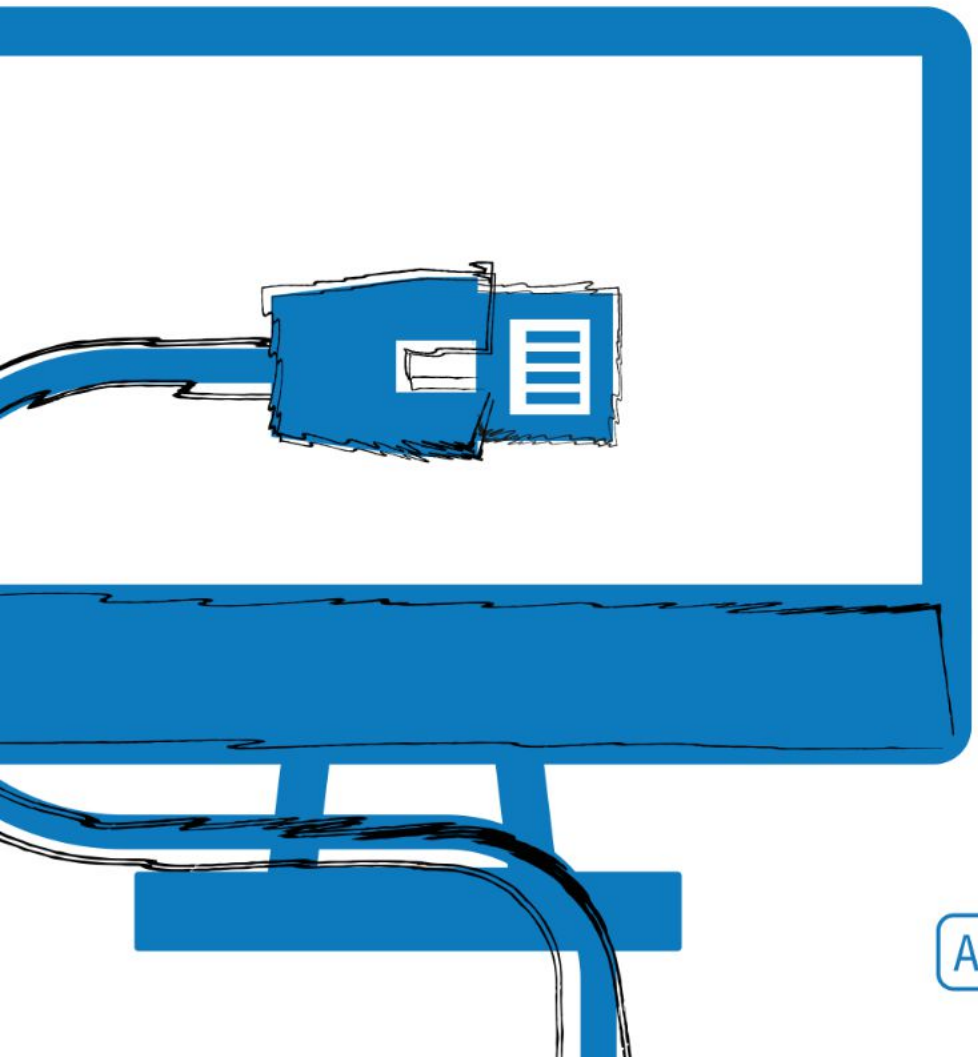
Примечание: виртуальный диск должен быть расположен в каталоге `/var/lib/libvirt/images`.

Конфигурация виртуальной среды

Имя ВМ	Назначение	IP адрес	Память (Мб)	VCPU
AL-1724V-host	Хост Docker	192.168.120.2	4096	2

Модуль 2

Установка и управление локальной виртуализацией QEMU/KVM



ASTRA LINUX®

Модуль 2. Установка и управление локальной виртуализацией QEMU/KVM

Описание практической работы

Общее время выполнения: 60 мин. (1 час)

- Задание 1. Установка виртуализации QEMU/KVM (10 мин.)
- Задание 2. Настройка предварительных требований (15 мин.)
- Задание 3. Развертывание виртуальной машины (10 мин.)
- Задание 4. Управление виртуальной машиной (25 мин.)

Задание 1. Установка виртуализации QEMU/KVM

Оценка времени выполнения: 10 мин.

1 Выполнить вход на хост под учетной записью локального администратора.

2 Выполнить проверку поддержки аппаратной виртуализации процессорными ядрами хоста:

```
grep -E 'svm|vmx' /proc/cpuinfo
```

Примечание: аппаратная виртуализация поддерживается если в выводе flags присутствует инструкция svm или vmx.

```
sa@astra:~$ grep -E 'svm|vmx' /proc/cpuinfo
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall
tant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf tsc_known_freq pni pclmulqdq dtes64 monitor ds_cpl vmx smx
x16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb invpci
tibp ibrs_enhanced tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid rdseed adx smap clflushopt cl
pt xsavec xgetbv1 xsaves split_lock_detect avx_vnni dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp hwp_pkg_req umip pku ospke waitp
me rdpid movdiri movdir64b fsrm md_clear serialize pconfig arch_lbr flush_l1d arch_capabilities
```

```
lsmod | grep kvm
```

Примечание: в выводе команды, kvm_intel или kvm_amd указывает на то, что модуль ядра поддержки аппаратной виртуализации загружен.

```
sa@astra:~$ lsmod | grep kvm
kvm_intel             339968  54
kvm                  1011712  1 kvm_intel
```

3 Выполнить проверку версий ядра и ОС:

```
uname -r
lsb_release -d
```

Примечание: вывод команд должен указывать на современную версию ядра Linux (новее версии 2.6.2).

4 Установить виртуализацию QEMU/KVM:

```
sudo apt install astra-kvm
```

5 Перезагрузить хост:

```
sudo reboot
```

6 Проверить работоспособность виртуализации:

Примечание: демон **libvirtd** должен быть загружен и активен и тесты проверки виртуализации не должны содержать ошибок.

```
sudo systemctl status libvirtd.service
```

```
sa@astra:~$ sudo systemctl status libvirtd.service
● libvirtd.service - Virtualization daemon
   Loaded: loaded (/lib/systemd/system/libvirtd.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2023-07-09 14:34:57 +05; 5h 21min ago
     Docs: man:libvirtd(8)
           https://libvirt.org
  Process: 2596 ExecStartPre=/usr/sbin/pdp-init-libvirt (code=exited, status=0/SUCCESS)
  Process: 2604 ExecStartPre=/usr/sbin/pdp-init-extstorage (code=exited, status=0/SUCCESS)
```

```
sudo virt-host-validate qemu
```

7 Проверить создание виртуальной сети default:

```
virsh -c qemu:///system net-list --all
```

Задание 2. Настройка предварительных требований

Оценка времени выполнения: 15 мин.

1 Настроить административный доступ:

```
sudo usermod -a -G kvm,libvirt,libvirt-qemu,libvirt-admin
<имя учетной записи администратора>
```

2 Создать файл /tmp/nat-network.xml:

```
touch /tmp/nat-network.xml
```

3 Создать виртуальную сеть nat-network:

примечание: вместо <Z> следует указать следующий доступный, после существующего, номер виртуального коммутатора (например 1).

```
cat > /tmp/nat-network.xml <<EOF
<network>
  <name>nat-network</name>
    <forward dev="eth0" mode="nat">
    <nat>
      <port start="1024" end="65535"/>
    </nat>
    <interface dev="eth0"/>
  </forward>
  <bridge name="virbr<Z>" stp="on" delay="0"/>
  <domain name="nat-network"/>
  <ip address="192.168.100.1" netmask="255.255.255.0">
  </ip>
</network>
EOF
```

4 Запустить виртуальную сеть nat-network:

```
virsh -c qemu:///system net-start nat-network
```

5 Настроить автоматический запуск виртуальной сети nat-network:

```
virsh -c qemu:///system net-autostart nat-network
```

6 Проверить создание виртуального коммутатора virbr<Z>:

```
sudo brctl show
```

```
sa@astra:~$ sudo brctl show
bridge name     bridge id               STP enabled
docker0         8000.0242fc834498       no
virbr0          8000.5254002fc8cb       yes
virbr1          8000.5254001b0f47       yes
virbr2          8000.52540048a31e       yes
```

7 Создать и настроить каталог пула хранения pool1:

```
sudo mkdir -p /lab/pool1
sudo chmod 777 /lab/pool1
sudo chown <имя локального администратора>:<имя локального администратора> /lab/pool1
```

8 Создать и настроить пул хранения pool1:

```
virsh -c qemu:///system pool-define-as pool1 --type dir --target /lab/pool1
virsh -c qemu:///system pool-build pool1
virsh -c qemu:///system pool-start pool1
virsh -c qemu:///system pool-autostart pool1
```

9 Проверить параметры пула хранения:

```
virsh -c qemu:///system pool-list --all
virsh -c qemu:///system pool-info pool1
```

примечание: пул хранения должен быть активен и запускаться автоматически при старте ОС.

```
sa@astra:~$ virsh -c qemu:///system pool-list --all
 Имя       Состояние   Автозапуск
-----------------------------------
 default   активен     yes
 pool1     активен     yes

sa@astra:~$ virsh -c qemu:///system pool-info pool1
Имя:          pool1
UUID:           be7c5250-e920-4d49-9512-ab1fb4382eda
Состояние: работает
Постоянство: yes
Автозапуск: yes
Размер:   914,38 GiB
Выделение: 594,72 GiB
Доступно: 319,67 GiB
```

Задание 3. Развертывание виртуальной машины

Оценка времени выполнения: 10 мин.

1 Скопировать диск виртуальной машины *AL-1724V-template*:

```
sudo cp /var/lib/libvirt/images/AL-1724V-template.qcow2
/lab/pool1/
```

2 Выполнить импорт виртуальной машины *AL-1724V-template*:

```
virt-install --connect qemu:///system --name AL-1724V-
template --disk path=/lab/pool1/AL-1724V-template.qcow2
,cache=none --import --vcpus 2 --memory 8192 --video virtio --
os-variant=debian10 --network network=default --graphics=spice
```

3 Выполнить вход в виртуальную машину *AL-1724V-template* под учетной записью **astra** и паролем **astra**.

4 Завершить работу виртуальной машины *AL-1724V-template*:

```
sudo poweroff
```

Задание 4. Управление виртуальной машиной

Оценка времени выполнения: 25 мин.

1 Подготовить диск виртуальной машины *AL-1724V-template* к клонированию:

```
sudo virt-sysprep -c qemu:///system -a /lab/pool1/AL-1724V-template.qcow2
```

2 Выполнить клонирование виртуальной машины *AL-1724V-template*:

```
virt-clone --connect qemu:///system --original AL-1724V-template --name AL-1724V-host --file /lab/pool1/AL-1724V-host.qcow2
```

3 Настроить виртуальную машину *AL-1724V-host*:

- выполнить вход под учетной записью **astra** и паролем **astra**.
- изменить имя узла:

  ```
  sudo hostnamectl set-hostname dockerhost
  ```
- настроить IPv4 параметры сетевого интерфейса *net1*:

  Метод: Вручную

  Адрес: `192.168.120.2`

  Маска сети: `255.255.255.0`

  Шлюз: `192.168.120.1`

  Серверы DNS: `8.8.8.8`
- активировать интерфейс *net1*:

  ```
  nmcli con up net1
  ```
- настроить файл *hosts*:

  ```
  sudo nano /etc/hosts
  ```
- изменить имя и IP-адрес узла:

  ```
  192.168.120.2    dockerhost
  ```
- выполнить реконфигурацию ключей SSH виртуальной машины:

  ```
  sudo dpkg-reconfigure openssh-server
  ```
- завершить работу виртуальной машины *AL-1724V-host*:

  ```
  sudo poweroff
  ```

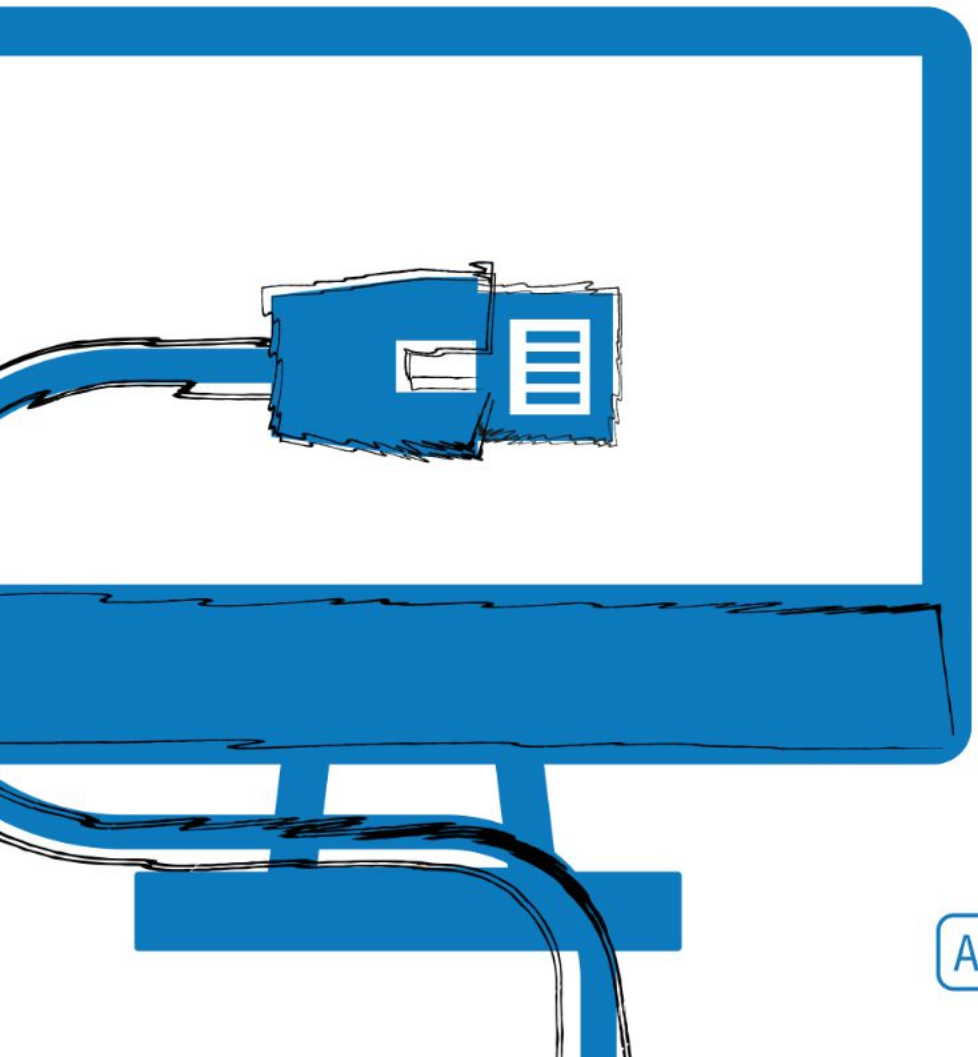
4 Создать снимок состояния виртуальной машины *AL-1724V-host*:

```
virsh -c qemu:///system snapshot-create AL-1724V-host
```

5 Запустить виртуальную машину *AL-1724V-host*.

Модуль 3

Установка и управление контейнерной виртуализацией



ASTRA LINUX®

Модуль 3. Установка и управление контейнерной виртуализацией

Описание практической работы

Общее время выполнения: 120 мин. (2 часа)

- Задание 1. Установка Docker в ОС Astra Linux (15 мин.)
- Задание 2. Установка и управление образами Docker (15 мин.)
- Задание 3. Запуск и управление контейнерами Docker (30 мин.)
- Задание 4. Создание образов Docker (30 мин.)
- Задание 5. Создание и запуск многоконтейнерного приложения (15 мин.)
- Задание 6. Загрузка образа на Docker Hub (15 мин.)

Задание 1. Установка Docker в ОС Astra Linux

Оценка времени выполнения: 15 мин.

1 Выполнить вход на виртуальную машину *AL-1724V-host* под учетной записью **astra** и паролем **astra**.

2 Установить пакет *docker.io*:
```
sudo apt install -y docker.io
```

3 Добавить администратора *astra* в группу *docker*:
```
sudo usermod -aG docker astra
```

4 Установить пакет *rootless-helper-astra*:
```
sudo apt install rootless-helper-astra
```

5 Запустить демон Docker от имени пользователя *astra*:
```
sudo systemctl start rootless-docker@astra
sudo systemctl enable rootless-docker@astra
```

6 Скачать .deb пакет Visual Studio Code:
- перейти по ссылке в браузере Firefox https://code.visualstudio.com/
- нажать на кнопку «.deb»

7 Установить пакет code_<версия сборки>_amd64.deb:

```
sudo apt install /home/astra/Загрузки/code_<версия сборки>_amd64.deb
```

8 Установить плагин *docker-compose*:

```
DOCKER_CONFIG=${DOCKER_CONFIG:-$HOME/.docker}
mkdir -p $DOCKER_CONFIG/cli-plugins
curl -SL https://github.com/docker/compose/releases/download/v2.18.1/docker-compose-linux-x86_64 -o $DOCKER_CONFIG/cli-plugins/docker-compose
chmod +x $DOCKER_CONFIG/cli-plugins/docker-compose
```

9 Включить запуск контейнеров Docker в режиме пониженной целостности:

```
sudo astra-docker-isolation enable
sudo systemctl restart docker
```

10 Проверить установку docker и docker-compose:

```
rootlessenv docker version
```

```
astra@dockerhost:~$ rootlessenv docker version
Client:
 Version:           20.10.2+dfsg1
 API version:       1.41
 Go version:        go1.15.9
 Git commit:        2291f61
 Built:             Thu Apr  6 14:43:28 2023
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server:
 Engine:
  Version:          20.10.2+dfsg1
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.15.9
  Git commit:       8891c58
  Built:            Thu Apr  6 14:43:28 2023
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.4.13~ds1
  GitCommit:        1.4.13~ds1-1~deb11u2astra.se2+ci2
 runc:
  Version:          1.1.4+ds1
  GitCommit:        1.1.4+ds1-1
 docker-init:
  Version:          0.18.0
  GitCommit:
```

```
rootlessenv docker compose version
```

```
astra@dockerhost:~$ rootlessenv docker compose version
Docker Comoose version v2.18.1
```

11 Завершить работу виртуальной машины *AL-1724V-host*.

12 Удалить снимок состояния виртуальной машины *AL-1724V-host*:

```
virsh -c qemu:///system snapshot-delete AL-1724V-host --
current
```

13 Создать снимок состояния виртуальной машины *AL-1724V-host*:

```
virsh -c qemu:///system snapshot-create AL-1724V-host
```

14 Запустить виртуальную машину *AL-1724V-host*.

Задание 2. Установка и управление образами Docker

Оценка времени выполнения: 15 мин.

1. Выполнить вход на виртуальную машину *AL-1724V-host* под учетной записью **astra** и паролем **astra**.
2. Скачать образы из реестра Docker Hub:

```
rootlessenv docker pull hello-world:latest
rootlessenv docker pull python
rootlessenv docker pull nginx
```

3. Получить список установленных образов:

```
rootlessenv docker images
```

4. Получить детальную информацию об образе *hello-world*:

```
rootlessenv docker image inspect hello-world
```

Задание 3. Запуск и управление контейнерами Docker

Оценка времени выполнения: 30 мин.

1. Создать и запустить контейнер *hello-world* с именем **test**:
   ```
   rootlessenv docker run --name test hello-world
   ```
2. Получить список всех контейнеров хоста:
   ```
   rootlessenv docker ps -a
   ```
3. Повторно запустить контейнер *test*:
   ```
   rootlessenv docker start -a test
   ```
4. Создать и запустить контейнер nginx в фоновом режиме, с подключением псевдотерминала TTY:
   ```
   rootlessenv docker run --name nginx-test -d nginx
   ```
5. Получить информацию об IP-адресе контейнера:
   ```
   rootlessenv docker container inspect nginx-test | grep IPAddress
   ```
6. Запустить оболочку bash в контейнере nginx-test:
   ```
   rootlessenv docker exec -it nginx-test bash
   ```
7. Запустить команды:
   ```
   ls
   hostname
   hostname -i
   ```
8. Выполнить выход из оболочки, нажав **CTRL-D**.
9. Опубликовать внутренний порт TCP 80 в новом контейнере nginx в фоновом режиме:
   ```
   rootlessenv docker run --name nginx-test2 -d -p 8080:80 nginx
   ```
10. Убедиться в том, что порт TCP 80 контейнера **`nginx-test2`** доступен с хоста, открыв браузер Firefox и перейдя по ссылке: http://localhost:8080/

11 Подключить каталог */home/astra/Загрузки* к каталогу */tmp* в новом контейнере nginx, в фоновом режиме совместно, с параметром автоматического удаления контейнера после его остановки:

```
rootlessenv docker run -v /home/astra/Загрузки:/tmp --name nginx-test3 -d --rm nginx
```

12 Запустить оболочку bash в контейнере *nginx-test3* и убедиться в том, что содержимое каталога */home/astra/Загрузки* доступно в каталоге */tmp*:

```
rootlessenv docker exec -it nginx-test bash
cd tmp
ls
```

13 Выполнить выход из оболочки, нажав **CTRL-D**.

14 Остановить работу контейнера *nginx-test3*:

```
rootlessenv docker container stop nginx-test3
```

15 Убедиться в том, что контейнер *nginx-test3* был удален:

```
rootlessenv docker ps -a
```

16 Удалить запущенный контейнер *nginx-test2*:

```
rootlessenv docker rm -f nginx-test2
```

17 Удалить все остановленные контейнеры хоста:

```
rootlessenv docker container prune
```

18 Удалить все контейнеры хоста:

```
rootlessenv docker container prune
```

Задание 4. Создание образов Docker

Оценка времени выполнения: 30 мин.

1 Создать и настроить каталог */lab*:

```
sudo mkdir -p /lab
sudo chmod 777 /lab
sudo chown astra:astra /lab
```

2 Создать каталог */server*:

```
mkdir -p /lab/server
```

3 Создать файлы сервиса *server*:

```
touch /lab/server/index.html
touch /lab/server/server.py
touch /lab/server/Dockerfile
```

4 Запустить Visual Studio Code:

```
code
```

5 Установить расширения Python, Docker и Docker Extension Pack:

- нажать CTRL-SHIFT-X;
- в секции «Popular” нажать кнопку «Install» в расширении «Python»;
- в строке поиска набрать docker;

6 Открыть файл *server.py* для редактирования в Visual Studio Code и добавить в него следующий код:

```
import http.server
import socketserver

handler = http.server.SimpleHTTPRequestHandler
with socketserver.TCPServer(("", 1234), handler) as httpd:

   httpd.serve_forever()
```

7 Сохранить файл, нажав CTRL-S.

8 Открыть файл *index.html* для редактирования в Visual Studio Code и добавить в него следующий текст:

```
Работает супер сервер!
```

9 Сохранить файл, нажав CTRL-S.

10 Открыть файл Dockerfile сервиса server для редактирования в Visual Studio Code и добавить в него следующий код:

```
FROM python:latest
WORKDIR /server
COPY . .
CMD ["python","server.py"]
```

11 Сохранить файл, нажав CTRL-S.

12 Создать каталог */client:*

```
mkdir -p /lab/client
```

13 Создать файл сервиса *client*:

```
touch /lab/client/client.py
touch /lab/client/Dockerfile
```

14 Открыть файл *client.py* для редактирования в Visual Studio Code и добавить в него следующий код:

```
import urllib.request
fp = urllib.request.urlopen("http://localhost:1234/")

encodedContent = fp.read()
decodedContent = encodedContent.decode("utf8")

print(decodedContent)
fp.close()
```

15 Сохранить файл, нажав CTRL-S.

16 Открыть файл *Dockerfile* сервиса *client* для редактирования в Visual Studio Code и добавить в него следующий код:

```
FROM python:latest
WORKDIR /client
COPY . .
CMD ["python","client.py"]
```

17 Сохранить файл, нажав CTRL-S.

18 Создать образ сервиса клиента *client-image* с тэгом **1.0.0**:

```
cd /lab/client
rootlessenv docker build . -t  client-image:1.0.0
```

19 Создать образ сервиса сервер *server-image* с тэгом **1.0.0**:

```
cd /lab/server
rootlessenv docker build . -t  server-image:1.0.0
```

20 Создать и запустить контейнер сервиса *server* в фоновом режиме:

```
rootlessenv docker run --name server-service --rm -d
server-image:1.0.0
```

21 Убедиться в том, что контейнер server запущен:

```
rootlessenv docker ps -a
```

22 Удалить запущенный контейнер *server*:

```
rootlessenv docker rm -f server-service
```

Задание 5. Создание и запуск многоконтейнерного приложения

Оценка времени выполнения: 15 мин.

1 Создать файл Docker Compose для сборки многоконтейнерного приложения:
```
touch /lab/compose.yaml
```

2 Открыть файл compose.yaml для редактирования в Visual Studio Code и добавить в него следующий код:
```
services:

  server-service:
    image: server-image:1.0.0
    ports:
      - 1234:1234

  client-service:
    image: client-image:1.0.0
    network_mode: host
    depends_on:
      - server-service
```

3 Сохранить файл, нажав CTRL-S.

4 Создать и запустить многоконтейнерное приложение из сервисов *server-service* и *client-service*:
```
cd /lab
rootlessenv docker-compose build
rootlessenv docker-compose up
```

5 Остановить многоконтейнерное приложение, нажав CTRL-C.

6 Просмотреть размер объектов Docker:
```
rootlessenv docker system df
```

```
astra@dockerhost:/lab$ rootlessenv docker system df
TYPE            TOTAL     ACTIVE    SIZE      RECLAIMABLE
Images          5         2         1.195GB   1.195GB (99%)
Containers      2         0         3.556MB   3.556MB (100%)
Local Volumes   0         0         0B        0B
Build Cache     0         0         0B        0B
```

Задание 6. Загрузка образа на Docker Hub

Оценка времени выполнения: 15 мин.

1. Зарегистрируйте вашу личную учетную запись на ресурсе DockerHub:
2. запустите браузер Firefox и перейдите по ссылке: https://hub.docker.com/
3. укажите имя пользователя в поле «Username»;
4. укажите адрес электронной почты в поле «Email»;
5. укажите пароль вашей учетной записи в поле «Password»;
6. отметьте галочкой «I agree to the Subscription Service Agreement, Privacy Policy, and Data Processing Terms»;
7. пройдите проверку CAPTCHA отметив галочкой «Я не робот»;
8. нажмите на кнопку «Sign Up».
9. Перейдите на страницу «Repositories».
10. Нажмите на кнопку «Create Repository».
11. Укажите имя репозитория (например demo), выберите тип репозитория «Private» и нажмите на кнопку «Create repository».
12. Выполните авторизацию в DockerHub указав ваше имя пользователя и пароль:

    ```
    rootlessenv docker login
    ```

13. Настройте тэг личного репозитория для образа **client-image**:

    ```
    rootlessenv docker tag client-image:1.0.0 <имя личной учетной записи на DockerHub>/<имя личного репозитория>:latest
    ```

14. Загрузите образ **client-image** в личный репозиторий DockerHub:

    ```
    rootlessenv docker push <имя личной учетной записи на DockerHub>/<имя личного репозитория>:latest
    ```

15. Выполните удаление всех объектов Docker на хосте:

    ```
    rootlessenv docker system prune
    ```

16. Загрузите образ **client-image** из личного репозитория DockerHub:

    ```
    rootlessenv docker pull <имя личной учетной записи на DockerHub>/<имя личного репозитория>
    ```