Московский государственный технический университет
им. Н. Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5. Курс «Автоматизация разработки и эксплуатации ПО»

Отчет по лабораторной работе №1

«Введение в Linux»

Выполнил:

Студентка группы ИУ5-71Б

Ноздрова Валентина

Дата: 17.09.2022

Подпись:

Проверил:

Дата:

Подпись:

Москва, 2022 г.

План и задачи лабораторной работы:

Часть 1. Начало работы

1. Подготовка рабочего окружения
2. Права, пользователи, su и sudo
3. Настройка сети ВМ
4. Подключение по ssh, ssh-agent
5. Установка и работа с tmux
6. Выполнение базовых команд

Часть 2. Продвинутая работа с системой

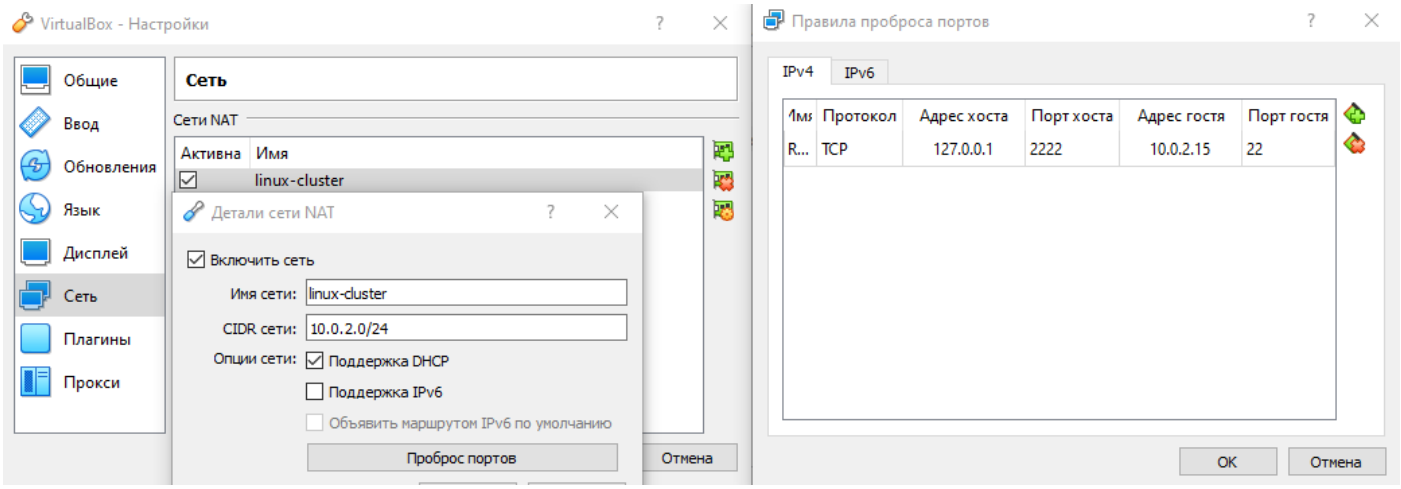
1. Автоматизируем сбор данных о системе с bash
2. Делаем сбор данных регулярным с cron
3. Пишем свой systemd-сервис
4. Запускаем процесс внутри нового пространства имен
5. Установка docker и запуск hello-world

Ход выполнения работы:

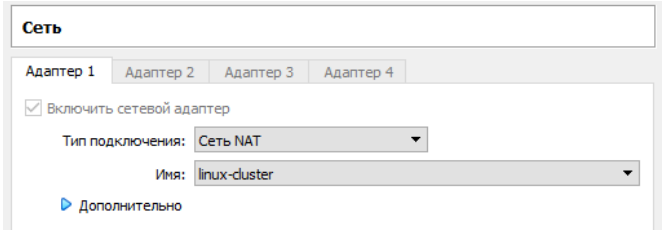
1.1 Подготовка рабочего окружения

Установим VirtualBox и создаем виртуальную машину. При установке укажем использование динамического виртуального диска.

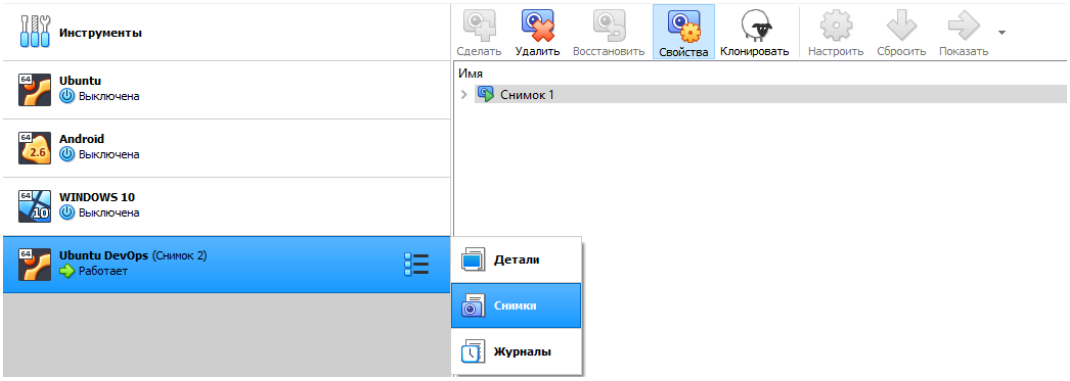
В настройках VirtualBox создадим новую сеть NAT *linux-cluster*. Настроим проброс портов 127.0.0.1:2222 -> 10.0.2.15:22.



В настройках ВМ в пункте «Сеть» выбираем «Сеть NAT» и новую сеть.



В настройках ВМ в системе виртуализации выбираем загруженный образ в качестве содержимого оптического носителя, чтобы с него загрузиться. Запускаем виртуальную машину и устанавливаем систему. В процессе установки выбираем установку OpenSSH-сервера. После установки делаем мгновенный снимок виртуальной машины.



1.2 Права, пользователи, su и sudo

Станем суперпользователем с помощью утилиты *sudo –i*.

```
user@devopsiu5:~$ sudo -i
[sudo] password for user:
root@devopsiu5:~# _
```

С помощью утилиты *visudo* отредактируем файл */etc/sudoers*, добавив в него строку:

```
%wheel ALL=(ALL;ALL) ALL
```

Данная строка означает, что пользователи группы *wheel* могут запускать все команды от лица всех пользователей и всех групп на всех хостах.

```
 GNU nano 4.8                        /etc/sudoers.tmp                      Modified
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
%wheel ALL=(ALL;ALL) ALL


^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos   M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line M-E Redo
```

Добавим пользователя *ansible* с домашней директорией */home/ansible* с помощью утилиты *adduser*.

```
root@devopsiu5:~# adduser ansible
Adding user `ansible' ...
Adding new group `ansible' (1001) ...
Adding new user `ansible' (1001) with group `ansible' ...
Creating home directory `/home/ansible' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for ansible
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
```

С помощью команды *passwd ansible* изменим пароль пользователя *ansible*.

```
root@devopsiu5:~# passwd ansible
New password:
Retype new password:
passwd: password updated successfully
root@devopsiu5:~# _
```

Аналогичным способом изменим пароль пользователя *root*.

```
root@devopsiu5:~# passwd root
New password:
Retype new password:
passwd: password updated successfully
```

Создадим группу *wheel* с помощью команды *groupadd wheel*. Создадим директорию */admin* с помощью *mkdir /admin*. Сделаем владельцем этой директории пользователя *ansible* и выдадим права группе *wheel* с помощью команды *chown ansible:wheel /admin*. Выдадим права на чтение и редактирование пользователю *ansible*, а группе *wheel* только на чтение с помощью команды *chmod 640 /admin*.

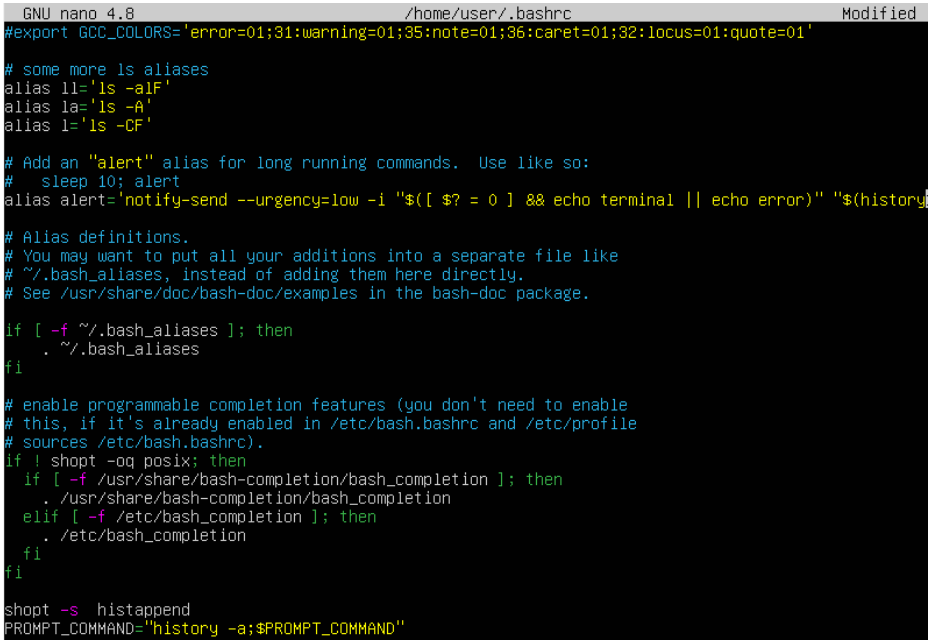
```
root@devopsiu5:~# groupadd wheel
root@devopsiu5:~# mkdir /admin
root@devopsiu5:~# chown ansible:wheel /admin
root@devopsiu5:~# chmod 640 /admin
```

Настроим для *root* сохранение истории, добавив в конец */root/.bashrc*:

```
| shopt -s histappend

| PROMPT_COMMAND="history -a;$PROMPT_COMMAND"
```

```
  GNU nano 4.8                          /root/.bashrc                          Modified
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dircolors -b)"
    alias ls='ls --color=auto'
    #alias dir='dir --color=auto'
    #alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi

# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
#if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
#    . /etc/bash_completion
#fi
shopt -s histappend
PROMPT_COMMAND="history -a;$PROMPT_COMMAND"
```

Аналогичное действие проведем для пользователя *user*.



Для синхронизации *history* между сессиями выполним команду:

```
root@devopsiu5:~# export PROMPT_COMMAND='history -a; history -c; history -r'
```

Команда внутри кавычек добавится в качестве значения переменной окружения *PROMPT_COMMAND.*

Выйдем из пользователя *root* с помощью *^D*.

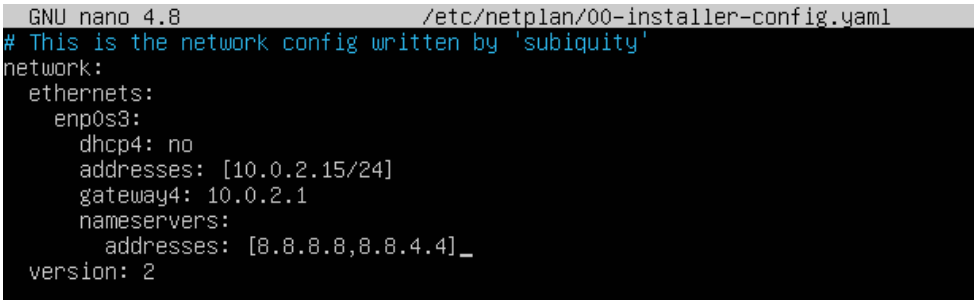
1.3. Настройка сети ВМ

Посмотрим сетевые интерфейсы и списки маршрутов:

```
user@devopsiu5:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
0
    link/ether 08:00:27:fb:13:23 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.4/24 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 542sec preferred_lft 542sec
    inet6 fe80::a00:27ff:fefb:1323/64 scope link
       valid_lft forever preferred_lft forever
```

```
user@devopsiu5:~$ ip r
default via 10.0.2.1 dev enp0s3 proto dhcp src 10.0.2.4 metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.4
10.0.2.1 dev enp0s3 proto dhcp scope link src 10.0.2.4 metric 100
```

Отредактируем файл сетевых настроек */etc/netplan/00-installer-config.yaml*.



Применим измененные настройки:

```
user@devopsiu5:~$ sudo netplan --debug apply
```

1.4. Подключение по ssh, ssh-agent

На основную систему установим ssh-клиент PuTTY.

На ВМ выполним команду *systemctl status sshd*, чтобы проверить, что *ssh*-сервис запущен.

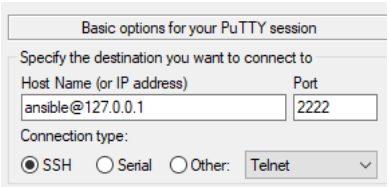
```
user@devopsiu5:~$ systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2022-09-16 21:07:53 UTC; 3 days ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 1344 (sshd)
      Tasks: 1 (limit: 1066)
     Memory: 1.1M
     CGroup: /system.slice/ssh.service
             └─1344 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

Sep 16 21:07:53 devopsiu5 systemd[1]: ssh.service: Succeeded.
Sep 16 21:07:53 devopsiu5 systemd[1]: Stopped OpenBSD Secure Shell server.
Sep 16 21:07:53 devopsiu5 systemd[1]: Starting OpenBSD Secure Shell server...
Sep 16 21:07:53 devopsiu5 sshd[1344]: Server listening on 0.0.0.0 port 22.
Sep 16 21:07:53 devopsiu5 sshd[1344]: Server listening on :: port 22.
Sep 16 21:07:53 devopsiu5 systemd[1]: Started OpenBSD Secure Shell server.
```

В утилите PuTTYgen сгенерируем ключ и сохраним публичный ключ. На ВМ сохраним публичную часть ключа в */home/ansible/.ssh/authorized_keys*.

Проверим права файла */home/ansible/.ssh/authorized_keys*:

```
user@devopsiu5:~$ ls -l /home/ansible/.ssh/authorized_keys
-rw-r--r-- 1 root root 398 Sep 16 22:33 /home/ansible/.ssh/authorized_keys
```

Подключимся по ssh (PuTTY+pageant для windows) под пользователем *ansible* и станем *root*-ом с помощью *sudo*. Подключимся на проброс портов: @127.0.0.1:2222.



Разрешим для пользователя *ansible* команды *sudo*, например, добавив в группу *wheel*.

```
user@devopsiu5:~$ sudo usermod -a -G wheel ansible
```

Подключимся к ssh-сессии:

```
root@devopsiu5: ~
Using username "ansible".
ansible@127.0.0.1's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-125-generic x86_64)
```

```
ansible@devopsiu5:~$ sudo -i
[sudo] password for ansible:
root@devopsiu5:~#
```

1.5. Установка и работа с tmux

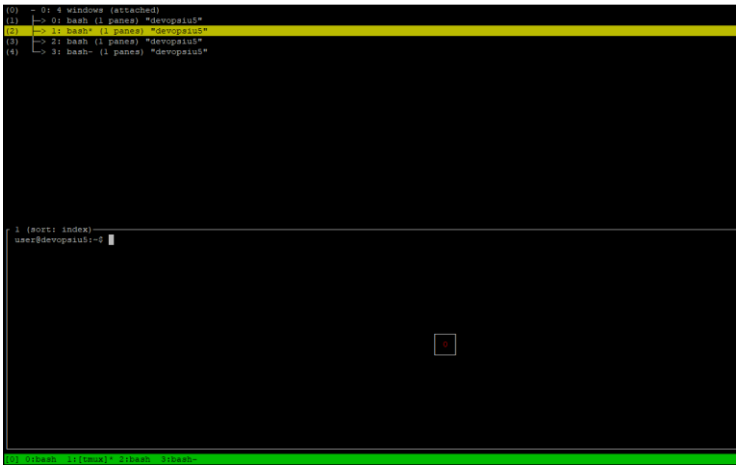
Установим утилиту *tmux* с помощью команды *sudo apt install tmux*.

```
user@devopsiu5:~$ sudo apt install tmux
Reading package lists... Done
Building dependency tree
Reading state information... Done
tmux is already the newest version (3.0a-2ubuntu0.3).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
```

Теперь запустим сессию с идентификатором по-умолчанию с помощью команды *tmux*. Создадим несколько окон с помощью ^B+C.

```
[0] 0:bash  1:bash  2:bash- 3:bash*
```

Переключимся между ними с помощью ^B+W.



1.6. Выполнение базовых команд

Выполним команду *man apt*:

```
APT(8)                                                    APT                                                    APT(8)

NAME
       apt - command-line interface

SYNOPSIS
       apt [-h] [-o=config string] [-c=config file] [-t=target release] [-a=architecture] {list | search | show | update |
           install pkg [{=pkg version number | /target release}]...  | remove pkg...  | upgrade | full-upgrade | edit-sources | {-v | --version} | {-h | --help}}

DESCRIPTION
       apt provides a high-level commandline interface for the package management system. It is intended as an end user interface and enables some options better
       suited for interactive usage by default compared to more specialized APT tools like apt-get(8) and apt-cache(8).

       Much like apt itself, its manpage is intended as an end user interface and as such only mentions the most used commands and options partly to not duplicate
       information in multiple places and partly to avoid overwhelming readers with a cornucopia of options and details.

       update (apt-get(8))
           update is used to download package information from all configured sources. Other commands operate on this data to e.g. perform package upgrades or
           search in and display details about all packages available for installation.

       upgrade (apt-get(8))
           upgrade is used to install available upgrades of all packages currently installed on the system from the sources configured via sources.list(5). New
           packages will be installed if required to satisfy dependencies, but existing packages will never be removed. If an upgrade for a package requires the
           removal of an installed package the upgrade for this package isn't performed.

       full-upgrade (apt-get(8))
           full-upgrade performs the function of upgrade but will remove currently installed packages if this is needed to upgrade the system as a whole.

       install, reinstall, remove, purge (apt-get(8))
           Performs the requested action on one or more packages specified via regex(7), glob(7) or exact match. The requested action can be overridden for
           specific packages by appending a plus (+) to the package name to install this package or a minus (-) to remove it.

           A specific version of a package can be selected for installation by following the package name with an equals (=) and the version of the package to
           select. Alternatively the version from a specific release can be selected by following the package name with a forward slash (/) and codename (buster,
           bullseye, sid ...) or suite name (stable, testing, unstable). This will also select versions from this release for dependencies of this package if
           needed to satisfy the request.

           Removing a package removes all packaged data, but leaves usually small (modified) user configuration files behind, in case the remove was an accident.
           Just issuing an installation request for the accidentally removed package will restore its function as before in that case. On the other hand you can
           get rid of these leftovers by calling purge even on already removed packages. Note that this does not affect any data or configuration stored in your
           home directory.

       autoremove (apt-get(8))
```

Пример скрипта, который создает файл с содержимым первого и второго аргументов:

```
user@devopsiu5:~$ nano file.sh
user@devopsiu5:~$ chmod +x file.sh
user@devopsiu5:~$ ./file.sh 10 15
user@devopsiu5:~$ cat test.txt
10
15
```

2.1. Автоматизируем сбор данных о системе с bash

Выведем LA:

```
user@devopsiu5:~$ cat /proc/loadavg | awk '{ print $1,$2,$3" processes: "$4", la
st PID: "$5
> }'
0.00 0.00 0.00 processes: 1/157, last PID: 16038
```

Выведем список сетевых интерфейсов в нужном формате:

```
user@devopsiu5:~$ ss -4tuln| awk  'NR>1{print $5}' | awk -F: '{print $1, $2}' | column -t
127.0.0.53%lo  53
127.0.0.53%lo  53
0.0.0.0        22
0.0.0.0        8000
```

Выведем список IP-адресов на хосте:

```
user@devopsiu5:~$ ip a | grep -E 'inet ' | awk '{ print $2 }'
127.0.0.1/8
10.0.2.15/24
```

Обернем это все в *bash*-скрипт */root/beholder.sh*

```
echo
date +"%d.%m.%Y %H:%M:%S"
echo
echo "LA: "
cat /proc/loadavg | awk '{ print $1, $2, $3" processes: "$4", last PID: "$5}'
echo
echo "Listening ports: "
ss -4tuln | awk 'NR>1{print $5}' | awk -F: '{print $1, $2}' | column -t
echo
echo "IP-addresses: "
ip a | grep -E 'inet ' | awk '{print $2 }'
echo
echo "Disk free: "
df -h
echo
echo "Inode free: "
df -i
echo
for i in $@
do
        echo "Next search by name:"
        ps aux | grep $i
        echo
done
echo "============================"
```

Запустим скрипт:

```
user@devopsiu5:~$ sudo /root/beholder.sh user

20.09.2022 16:57:17

LA:
0.00 0.00 0.00 processes: 1/159, last PID: 16206

Listening ports:
127.0.0.53%lo  53
127.0.0.53%lo  53
0.0.0.0        22
0.0.0.0        8000

IP-addresses:
127.0.0.1/8
10.0.2.15/24

Disk free:
Filesystem                         Size  Used Avail Use% Mounted on
udev                               445M     0  445M   0% /dev
tmpfs                               98M  1.1M   97M   2% /run
/dev/mapper/ubuntu--vg-ubuntu--lv  9.8G  4.6G  4.7G  50% /
tmpfs                              489M     0  489M   0% /dev/shm
tmpfs                              5.0M     0  5.0M   0% /run/lock
```

```
Inode free:
Filesystem                         Inodes  IUsed  IFree IUse% Mounted on
udev                               113743    484 113259    1% /dev
tmpfs                              125040    767 124273    1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv 655360 117380 537980   18% /
tmpfs                              125040      4 125036    1% /dev/shm
tmpfs                              125040      3 125037    1% /run/lock
tmpfs                              125040     18 125022    1% /sys/fs/cgroup
/dev/loop0                          11796  11796      0  100% /snap/core20/1611
/dev/loop1                            802    802      0  100% /snap/lxd/22753
/dev/loop2                            486    486      0  100% /snap/snapd/16292
/dev/sda2                          114688    316 114372    1% /boot
tmpfs                              125040     22 125018    1% /run/user/1000
tmpfs                              125040     22 125018    1% /run/user/1001

Next search by name:
user         944  0.0  0.8  19040  8660 ?        Ss   13:33   0:00 /lib/systemd/systemd --user
user         945  0.0  0.2 104056  2484 ?        S    13:33   0:00 (sd-pam)
user         950  0.0  0.4   8264  4016 ttyl     S    13:33   0:00 -bash
ansible     1583  0.0  0.8  19044  8700 ?        Ss   14:36   0:00 /lib/systemd/systemd --user
root        1769  0.0  0.3   8784  3736 pts/0    S    14:46   0:00 su user
user        1770  0.0  0.4   8264  4040 pts/0    S    14:46   0:00 bash
root        1787  0.0  0.3   8408  3516 pts/0    S    14:51   0:00 su user
user        1788  0.0  0.3   8264  3988 pts/0    S    14:51   0:00 bash
user        1895  0.0  0.3   7184  3608 pts/0    T    15:04   0:00 man awk
user        1904  0.0  0.1   7044  1040 pts/0    T    15:04   0:00 man awk
user        1905  0.0  0.2   5836  2360 pts/0    T    15:04   0:00 pager
root       10940  0.0  0.8  13920  8920 ?        Ss   16:11   0:00 sshd: user [priv]
user       13310  0.0  0.6  14052  6028 ?        S    16:11   0:00 sshd: user@pts/2
user       13311  0.0  0.5   8276  5048 pts/2    Ss   16:11   0:00 -bash
root       16202  0.0  0.4   9412  4712 pts/2    S+   16:57   0:00 sudo /root/beholder.sh user
root       16203  0.0  0.1   2608  1828 pts/2    S+   16:57   0:00 sh /root/beholder.sh user
root       16217  0.0  0.0   6300   720 pts/2    S+   16:57   0:00 grep user
```

2.2. Делаем сбор регулярным с cron

Впишем в cron новое правило и ждем некоторое время.

```
root@devopsiu5:~# sudo echo '* * * * * root /root/beholder.sh >> /tmp/beholder-output' > /etc/cron.d/beholder
```

Смотрим файл *cat /tmp/beholder-output*:

```
20.09.2022 17:07:01

LA:
0.00 0.00 0.00 processes: 2/159, last PID: 16369

Listening ports:
127.0.0.53%lo  53
127.0.0.53%lo  53
0.0.0.0        22
0.0.0.0        8000

IP-addresses:
127.0.0.1/8
10.0.2.15/24

Disk free:
Filesystem                         Size  Used Avail Use% Mounted on
udev                               445M     0  445M   0% /dev
tmpfs                               98M  1.1M   97M   2% /run
/dev/mapper/ubuntu--vg-ubuntu--lv  9.8G  4.6G  4.7G  50% /
tmpfs                              489M     0  489M   0% /dev/shm
tmpfs                              5.0M     0  5.0M   0% /run/lock
tmpfs                              489M     0  489M   0% /sys/fs/cgroup
/dev/loop0                          62M   62M     0 100% /snap/core20/1611
/dev/loop1                          68M   68M     0 100% /snap/lxd/22753
```

2.3 Пишем свой systemd-сервис

Создадим свой *systemd-unit*.

```
user@devopsiu5:~$ sudo nano /etc/systemd/system/myhttp.service
```

```
[Unit]
Description=MyHTTP Server
Documentation=http://example.org/
After=network.target

[Service]
WorkingDirectory=/root/www
ExecStart=/usr/bin/python3 -m http.server 8000
KillMode=mixed
Restart=on-failure
Type=simple

[Install]
WantedBy=multi-user.target
Alias=myhttpd.servicem
```

Создадим для него WorkingDirectory.

```
user@devopsiu5:~$ sudo mkdir -p /root/www
```

Перезагрузим конфигурацию *systemd*.

```
user@devopsiu5:~$ systemctl daemon-reload
==== AUTHENTICATING FOR org.freedesktop.systemd1.reload-daemon ===
Authentication is required to reload the systemd state.
Authenticating as: user
Password:
==== AUTHENTICATION COMPLETE ===
```

Запустим, добавим в автозапуск.

```
==== AUTHENTICATION COMPLETE ===
user@devopsiu5:~$ systemctl start myhttp
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===
Authentication is required to start 'myhttp.service'.
Authenticating as: user
Password:
==== AUTHENTICATION COMPLETE ===
user@devopsiu5:~$ systemctl enable myhttp
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-unit-files ===
Authentication is required to manage system service or unit files.
Authenticating as: user
Password:
==== AUTHENTICATION COMPLETE ===
Created symlink /etc/systemd/system/myhttpd.service → /etc/systemd/system/myhttp.s
ervice.
==== AUTHENTICATING FOR org.freedesktop.systemd1.reload-daemon ===
Authentication is required to reload the systemd state.
Authenticating as: user
Password:
==== AUTHENTICATION COMPLETE ===
```

Проверяем работу с помощью *curl* и *ss*.

```
user@devopsiu5:~$ curl http://127.0.0.1:8000/
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/stri
ct.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
</ul>
<hr>
</body>
</html>
user@devopsiu5:~$ ss -tl | grep 8000
LISTEN  0       5               0.0.0.0:8000            0.0.0.0:*
```

2.4 Запускаем процесс внутри нового пространства имен

Просмотрим список доступных пространств имен.

```
user@devopsiu5:~$ lsns
        NS TYPE   NPROCS   PID USER COMMAND
4026531835 cgroup      4   944 user /lib/systemd/systemd --user
4026531836 pid         4   944 user /lib/systemd/systemd --user
4026531837 user        4   944 user /lib/systemd/systemd --user
4026531838 uts         4   944 user /lib/systemd/systemd --user
4026531839 ipc         4   944 user /lib/systemd/systemd --user
4026531840 mnt         4   944 user /lib/systemd/systemd --user
4026531992 net         4   944 user /lib/systemd/systemd --user
user@devopsiu5:~$ ip netns
```

Создадим сетевой namespace.

```
user@devopsiu5:~$ sudo ip netns add myhttp
Cannot create namespace file "/run/netns/myhttp": File exists
```

Теперь запустим внутри пару команд:

```
user@devopsiu5:~$ sudo ip netns exec myhttp ip link set dev lo up
[sudo] password for user:
user@devopsiu5:~$ sudo ip netns exec myhttp /usr/bin/python3 -m http.server 8080 &
[1] 3427
user@devopsiu5:~$ Traceback (most recent call last):
  File "/usr/lib/python3.8/runpy.py", line 194, in _run_module_as_main
    return _run_code(code, main_globals, None,
  File "/usr/lib/python3.8/runpy.py", line 87, in _run_code
    exec(code, run_globals)
  File "/usr/lib/python3.8/http/server.py", line 1294, in <module>
    test(
  File "/usr/lib/python3.8/http/server.py", line 1249, in test
    with ServerClass(addr, HandlerClass) as httpd:
  File "/usr/lib/python3.8/socketserver.py", line 452, in __init__
    self.server_bind()
  File "/usr/lib/python3.8/http/server.py", line 1292, in server_bind
    return super().server_bind()
  File "/usr/lib/python3.8/http/server.py", line 138, in server_bind
    socketserver.TCPServer.server_bind(self)
  File "/usr/lib/python3.8/socketserver.py", line 466, in server_bind
    self.socket.bind(self.server_address)
OSError: [Errno 98] Address already in use
```

Проверяем:

```
user@devopsiu5:~$ ss -tul4n
Netid  State   Recv-Q  Send-Q   Local Address:Port    Peer Address:Port  Process
udp    UNCONN  0       0        127.0.0.53%lo:53           0.0.0.0:*
tcp    LISTEN  0       4096     127.0.0.53%lo:53           0.0.0.0:*
tcp    LISTEN  0       128            0.0.0.0:22           0.0.0.0:*
tcp    LISTEN  0       5              0.0.0.0:8000         0.0.0.0:*
```

```
user@devopsiu5:~$ sudo ip netns exec myhttp ss -tul4n
Netid  State   Recv-Q  Send-Q   Local Address:Port    Peer Address:Port  Process
tcp    LISTEN  0       5              0.0.0.0:8080         0.0.0.0:*
```

Проверим:

```
user@devopsiu5:~$ sudo ip netns exec myhttp curl http://127.0.0.1:8080
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/stri
ct.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href=".bash_history">.bash_history</a></li>
<li><a href=".bashrc">.bashrc</a></li>
<li><a href=".local/">.local/</a></li>
<li><a href=".profile">.profile</a></li>
<li><a href=".ssh/">.ssh/</a></li>
<li><a href="beholder.sh">beholder.sh</a></li>
<li><a href="snap/">snap/</a></li>
<li><a href="www/">www/</a></li>
</ul>
<hr>
</body>
</html>
```

```
user@devopsiu5:~$ sudo ip netns exec myhttp curl http://127.0.0.1:8000
curl: (7) Failed to connect to 127.0.0.1 port 8000: Connection refused
```

```
user@devopsiu5:~$ curl http://127.0.0.1:8080
curl: (7) Failed to connect to 127.0.0.1 port 8080: Connection refused
user@devopsiu5:~$ curl http://127.0.0.1:8000
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/stri
ct.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
</ul>
<hr>
</body>
</html>
```

2.5. Установка docker и запуск hello-world

Установим docker

```
user@devopsiu5:~$ sudo systemctl status docker
[sudo] password for user:
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: >
     Active: active (running) since Tue 2022-09-20 19:29:05 UTC; 11min ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 17494 (dockerd)
      Tasks: 8
     Memory: 21.8M
     CGroup: /system.slice/docker.service
             └─17494 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/conta>
```

Запустим hello-world:

```
user@devopsiu5:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

3. History выполненных команд

visudo
adduser ansible
passwd
groupadd wheel
mkdir /admin
chown ansible:wheel /admin
chmod 640 /admin
nano /root/.bashrc
nano home/user/.bashrc
echo '* * * * * root /root/beholder.sh >> /tmp/beholder.output ' > /etc/cron.d/beholder
cat /tmp/beholder-output

ip a
ip r
sudo nano /etc/netplan/00-installer-config.yaml
sudo netplan –debug apply
systemctl status sshd
sudo mkdir /home/ansible/.ssh

```
sudo nano /home/ansible/.ssh/authorized_keys
ls –l /home/ansible/.ssh/authorized_keys
sudo usermod –a –G wheel ansible
sudo apt install tmux
tmux
man apt
nano file.sh
chmod +x file.sh
./file.sh 10 15
cat test.txt
cat /proc/loadavg | awk ‘{print $1,$2,$3” processes: “$4”, last PID: “$5}’
ss -4tuln|awk ‘NR>1{print $5}’ | awk –F: ‘{print $1, $2}’ | column –t
ip a | grep –E ‘inet ‘ | awk ‘{ print $2 }’
sudo nano /root/beholder.sh
sudo chmod +x /root/beholder.sh
sudo /root/beholder.sh user
sudo nano /etc/systemd/system/myhttp.services
sudo mkdir -p /root/www
systemctl daemon-reload
systemctl start myhttp
curl http://127.0.0.1:8000/
ss –tl | grep 8000
lsns
sudo ip netns add myhttp
sudo ip netns exec myhttp ip link set dev lo up
ip netns exec myhttp /usr/bin/python3 -m http.server 8080 &
ss -tul4n
ip netns exec myhttp ss -tul4n
ip netns exec myhttp curl http://127.0.0.1:8080
ip netns exec myhttp curl http://127.0.0.1:8000
curl http://127.0.0.1:8080
curl http://127.0.0.1:8000
sudo apt install apt-transport-https ca-certificates curl software-propertiescommon
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
focal stable"
sudo apt update
sudo apt install docker-ce
sudo systemctl status docker
docker run hello-world
```

4. Контрольные вопросы

4.1. Что такое Linux?

Linux – это ядро ОС.

4.2. Что делает каждая из команд, применяемых в лабораторной работе?

ls – показывает содержимое текущий директории или конкретной папке;

cd – позволяет сменить директорию;

cp – копирует объект;

mkdir – создает каталог;

cat – вывод объекта на экран;

less – утилита для просмотра больших текстовых файлов;

grep – фильтрация строк по подстрокам;

df – показывает данные по файловой системе;

chown – позволяет сменить владельца файла или каталога;

chmod – изменяет права на файл или каталог.

4.3. В чем разница между su и sudo?

Su – команда, которая позволяет пользователю войти в систему под другим именем, не завершая текущий сеанс. Sudo – программа для системного администрирования, позволяющая делегировать привилегированные ресурсы пользователям.

4.4. Что происходит в подробностях, когда вы вводите cat file.txt?

Определяется наличие разрешения на чтение у пользователя и наличие файла с названием file.txt в текущей директории, после чего в на экран выводится содержимое file.txt.

4.5. Что такое файловый дескриптор? Какие создаются по умолчанию?

Файловый дескриптор – целое неотрицательное число, которое ядро возвращает процессу, создавшему новый поток ввода-вывода. По умолчанию дескриптор 0 соответствует потоку стандартного ввода процесса (клавиатуре), 1 – потоку стандартного вывода (терминалу), 2 – потоку диагностики.

4.6. Как дописать в файл вывод команды? Как перезаписать файл? Как избавиться от вывода ошибок?

Дописать вывод команды в файл можно с помощью >>. Перезаписать файл можно с помощью >. Избавиться от вывода ошибок с помощью 2> /dev/null.

4.7. Как определяются права на файл? Как сделать файл исполняемым?

Существует три категории пользователей (владелец, группа, остальные) и три параметра доступа (чтение, запись, выполнение). Права доступа каждого из трех типов для каждой из трех групп и некоторые дополнительные параметры доступа хранятся в битовом поле индексного дескриптора. Сделать файл исполняемым можно с помощью команды chmod +x <имя файла>.

4.8. Как создать пользователя в GNU/Linux?

Создать пользователя можно с помощью утилиты adduser.

4.9. Как пользоваться man?

Команда man используется для получения руководства по какой-либо команде: man <имя команды>.

4.10. Как выдать права на sudo?

Выдать права на sudo можно, отредактировав файл /etc/sudoers, например, через утилиту visudo. При этом нужно добавить строку %group hosts=(users:groups) commands или user hosts=(users:groups) commands, где %group и user – группа или пользователь, которой выдаются права; hosts – хосты, к которым применяется правило; users и groups – пользователи и группы, от лица которых данный пользователь или группа могут выполнять команды; commands – команды, к которым применяется правило.

4.11. Зачем .bashrc файл?

.bashrc – скриптовый файл, который запускается при входе пользователя в систему.

4.12. Как посмотреть занятое место на диске?

С помощью команды df –h.

4.13. Как подключиться к серверу по ssh? Подробно о всех возможных проблемах.

Установить клиент ssh (в unix-системах установлен по умолчанию). На сервере установить и запустить sshd (ssh daemon). Сгенерировать ssh-ключи на машине, с которой выполняется подключение. Сохранить публичный ключ на сервере в /home/<user>/.ssh/authorized_keys. Подключиться к серверу через ssh-клиент.

4.14. Что такое публичный и приватный ключ? Какой оставляем себе, а какой копируем на сервер?

Публичный ключ используется для шифрования сообщений, которые можно расшифровать только приватным ключом. Себе оставляем приватный ключ, публичный ключ копируем на сервер.

4.15. Ssh-agent - зачем?

Ssh-agent хранит секретный ключ в памяти в незашифрованном виде, чтобы не вводить пароль при каждом подключении.

4.16. В чем разница между soft и hard лимитами?

Soft лимит может быть изменен пользователем, hard лимит устанавливается администратором и не может быть превышен пользователем.

4.17. Кто такой суперпользователь и что ему можно?

Пользователь с идентификатором 0, которому доступно выполнение любых команд.

4.18. Нужно ли делать резервные копии? А что еще нужно делать?

Резервные копии делать нужно. А еще нужно проверять их на работоспособность.

4.19. Что такое systemd, как происходит загрузка системы?

Systemd – процесс на вершине дерева процессов с PID=1, для которого остальные процессы являются дочерними. При загрузке компьютера происходит последовательная передача управления от системной прошивки к загрузчику, от него – к ядру. Ядро запускает планировщик и выполняет init или systemd, после чего ядро переходит в бездействие, пока не получит внешний вызов.

4.20. Как поставить пакет? Что такое apt?

Поставить пакет можно с помощью apt install <имя пакета>. Apt – программа для установки, обновления и удаления программных пакетов.