

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО»
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
Кафедра компьютерной инженерии и моделирования

ОТЧЕТ ПО ПРАКТИЧЕСКОМУ ЗАДАНИЮ №6
«Знакомство с Ansible»

Практическая работа
по дисциплине «Современные технологии программирования»
студента 1 курса группы ПИ-б-о-231(2)
Аметов Кемран Ленверович
направления подготовки 09.03.04 «Программная инженерия»

Симферополь, 2024

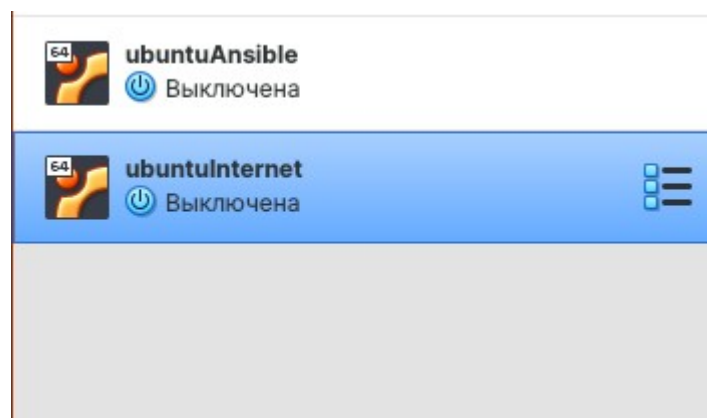
Цель:

Ознакомиться на практике с инструментом для удаленного управления конфигурациями Ansible.

Ход выполнения задания.

Подготовка виртуальных машин

1. Нам понадобится две чистые виртуальные машины. Если у вас дефицит дискового пространства, то ВМ из предыдущих работ можно удалить, здесь они нам не понадобятся. На одной из виртуальных машин будет размещаться ansible, поэтому я её так и назову - "Ansible", а вторая будет просто "коробкой" для других виртуальных машин, которыми мы будем управлять при помощи ansible, поэтому я назову её "Internet" (имена можете выбрать по своему вкусу). По итогу, с точки зрения сети, мы хотим получить такую топологию:
2. Создайте две виртуальные машины с Ubuntu Server . В процессе установки используйте стандартные настройки. Из дополнительного софта понадобится только ssh-сервер. Для машины с именем "Internet" создайте диск большого объёма 50+ГБ.
3. В настройках сети VirtualBox установите "Сетевой мост" для обеих машин.



Машина с Ansible

1. Для удобства дальнейшей работы, подключитесь к машине по ssh.

```
kemran@kemran:~$ ssh ansible@192.168.0.106
The authenticity of host '192.168.0.106 (192.168.0.106)' can't be established.
ED25519 key fingerprint is SHA256:b0Mr9ztX8y01z6GiAQjsXX60wc9xgtxh08qBeDSg0Mo.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.106' (ED25519) to the list of known hosts.
ansible@192.168.0.106's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-107-generic x86_64)

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

System information as of Thu May 16 12:14:22 PM UTC 2024

System load:  0.0               Processes:            108
Usage of /:   53.2% of 8.02GB   Users logged in:     1
Memory usage: 10%              IPv4 address for enp0s3: 192.168.0.106
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Thu May 16 12:10:21 2024
ansible@ansible:~$
```

2. Добавьте в список dns-серверов гугловский (8.8.8.8) и(или) яндексовый (77.88.8.8).

```
GNU nano 6.2 /etc/resolv.conf *
# This is /run/systemd/resolve/stub-resolv.conf managed by man:systemd-resolved(8).
# Do not edit.
#
# This file might be symlinked as /etc/resolv.conf. If you're looking at
# /etc/resolv.conf and seeing this text, you have followed the symlink.
#
# This is a dynamic resolv.conf file for connecting local clients to the
# internal DNS stub resolver of systemd-resolved. This file lists all
# configured search domains.
#
# Run "resolvectl status" to see details about the uplink DNS servers
# currently in use.
#
# Third party programs should typically not access this file directly, but only
# through the symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a
# different way, replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes of
# operation for /etc/resolv.conf.

nameserver 8.8.8.8
nameserver 77.88.8.8
nameserver 127.0.0.53
options edns0 trust-ad
search .

File Name to Write: /etc/resolv.conf
^G Help      M-D DOS Format  M-A Append     M-B Backup File
^C Cancel    M-M Mac Format  M-P Prepend    ^T Browse
```

3. Обновите индексы пакетов: `sudo apt-get update`.

```
ansible@ansible:~$ sudo apt-get update
Hit:1 http://ru.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://ru.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Hit:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:4 http://ru.archive.ubuntu.com/ubuntu jammy-backports InRelease
Fetched 119 kB in 1s (118 kB/s)
Reading package lists... Done
ansible@ansible:~$
```

4. Проверьте, что в системе установлен python 3: `python3 --version`. Если нет, установите.

```
ansible@ansible: $ python3 --version
Python 3.10.12
ansible@ansible: $
```

5. Проверьте, что в системе установлен pip: `python3 -m pip -V`. Если нет, установите.

```
ansible@ansible: $ python3 -m pip -V
/usr/bin/python3: No module named pip
ansible@ansible: $ sudo apt install pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'python3-pip' instead of 'pip'
The following additional packages will be installed:
  build-essential bzip2 cpp cpp-11 dpkg-dev fakeroot fontconfig-config fonts-dejavu-core g++ g++-11 gcc gcc-11
  gcc-11-base javascript-common libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan6
  libatomic1 libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libdeflate0 libdpkg-perl libexpat1-dev
  libfakeroot libfile-fcntllock-perl libfontconfig1 libgcc-11-dev libgd3 libgomp1 libisl23 libitm1 libjbig0
  libjpeg-turbo8 libjpeg8 libjs-jquery libjs-sphinxdoc libjs-underscore liblsan0 libmpc3 libnsl-dev libpython3-dev
  libpython3.10-dev libquadmath0 libstdc++-11-dev libtiff5 libtirpc-dev libtsan0 libubsan1 libwebp7 libxpm4
  linux-libc-dev lto-disabled-list make manpages-dev python3-dev python3-wheel python3.10-dev rpcsvc-proto
  zlib1g-dev
Suggested packages:
  bzip2-doc cpp-doc gcc-11-locales debian-keyring g++-multilib g++-11-multilib gcc-11-doc gcc-multilib autoconf
  automake libtool flex bison gdb gcc-doc gcc-11-multilib apache2 | lighttpd | httpd glibc-doc bzip libgd-tools
  libstdc++-11-doc make-doc
The following NEW packages will be installed:
  build-essential bzip2 cpp cpp-11 dpkg-dev fakeroot fontconfig-config fonts-dejavu-core g++ g++-11 gcc gcc-11
  gcc-11-base javascript-common libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan6
  libatomic1 libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libdeflate0 libdpkg-perl libexpat1-dev
  libfakeroot libfile-fcntllock-perl libfontconfig1 libgcc-11-dev libgd3 libgomp1 libisl23 libitm1 libjbig0
  libjpeg-turbo8 libjpeg8 libjs-jquery libjs-sphinxdoc libjs-underscore liblsan0 libmpc3 libnsl-dev libpython3-dev
  libpython3.10-dev libquadmath0 libstdc++-11-dev libtiff5 libtirpc-dev libtsan0 libubsan1 libwebp7 libxpm4
  linux-libc-dev lto-disabled-list make manpages-dev python3-dev python3-pip python3-wheel python3.10-dev
  rpcsvc-proto zlib1g-dev
0 upgraded, 64 newly installed, 0 to remove and 2 not upgraded.
Need to get 71.3 MB of archives.
After this operation, 239 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libc-dev-bin amd64 2.35-0ubuntu3.7 [20.3 kB]
Get:2 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 linux-libc-dev amd64 5.15.0-107.117 [1,333 kB]
Get:3 http://ru.archive.ubuntu.com/ubuntu jammy/main amd64 libcrypt-dev amd64 1:4.4.27-1 [112 kB]
Get:4 http://ru.archive.ubuntu.com/ubuntu jammy/main amd64 rpcsvc-proto amd64 1.4.2-0ubuntu6 [68.5 kB]
Get:5 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libtirpc-dev amd64 1.3.2-2ubuntu0.1 [192 kB]
Get:6 http://ru.archive.ubuntu.com/ubuntu jammy/main amd64 libnsl-dev amd64 1.3.0-2build2 [71.3 kB]
Get:7 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libc6-dev amd64 2.35-0ubuntu3.7 [2,100 kB]
Get:8 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 gcc-11-base amd64 11.4.0-1ubuntu1~22.04 [20.2 kB]
Get:9 http://ru.archive.ubuntu.com/ubuntu jammy/main amd64 libisl23 amd64 0.24-2build1 [727 kB]
Get:10 http://ru.archive.ubuntu.com/ubuntu jammy/main amd64 libmpc3 amd64 1.2.1-2build1 [46.9 kB]
Get:11 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 cpp-11 amd64 11.4.0-1ubuntu1~22.04 [10.0 MB]
Get:12 http://ru.archive.ubuntu.com/ubuntu jammy/main amd64 cpp amd64 4:11.2.0-1ubuntu1 [27.7 kB]
Get:13 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libcc1-0 amd64 12.3.0-1ubuntu1~22.04 [48.3 kB]
Get:14 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgomp1 amd64 12.3.0-1ubuntu1~22.04 [126 kB]
Get:15 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libitm1 amd64 12.3.0-1ubuntu1~22.04 [30.2 kB]
Get:16 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libatomic1 amd64 12.3.0-1ubuntu1~22.04 [10.4 kB]
```

6. Установите последнюю доступную версию ansible: `python3 -m pip install --user ansible`.

Флаг `--user` установит пакет ansible как локальный, т.е. ansible будет доступен только текущему пользователю.

```

ansible@ansible:~$ python3 -m pip install --user ansible
Collecting ansible
  Downloading ansible-9.5.1-py3-none-any.whl (47.0 MB)
    47.0/47.0 MB 4.4 MB/s eta 0:00:00
Collecting ansible-core~=2.16.6
  Downloading ansible_core-2.16.6-py3-none-any.whl (2.3 MB)
    2.3/2.3 MB 5.8 MB/s eta 0:00:00
Requirement already satisfied: PyYAML>=5.1 in /usr/lib/python3/dist-packages (from ansible-core~=2.16.6->ansible) (5.4.1)
Collecting resolvelib<1.1.0,>=0.5.3
  Downloading resolvelib-1.0.1-py2.py3-none-any.whl (17 kB)
Requirement already satisfied: Jinja2>=3.0.0 in /usr/lib/python3/dist-packages (from ansible-core~=2.16.6->ansible) (3.0.3)
Requirement already satisfied: cryptography in /usr/lib/python3/dist-packages (from ansible-core~=2.16.6->ansible) (3.4.8)
Collecting packaging
  Downloading packaging-24.0-py3-none-any.whl (53 kB)
    53.5/53.5 KB 9.9 MB/s eta 0:00:00
Installing collected packages: resolvelib, packaging, ansible-core, ansible
  WARNING: The scripts ansible, ansible-config, ansible-connection, ansible-console, ansible-doc, ansible-galaxy, ansible-inventory, ansible-playbook, ansible-pull and ansible-vault are installed in '/home/ansible/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
  WARNING: The script ansible-community is installed in '/home/ansible/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed ansible-9.5.1 ansible-core-2.16.6 packaging-24.0 resolvelib-1.0.1
ansible@ansible:~$

```

- После установки выскочит предупреждение, что запустить ansible по имени не получится, т.к. каталог в который он установился (~/.local/bin) не добавлен в PATH.

Нам не придётся делать это вручную, т.к. путь ~/.local/bin будет добавлен в PATH автоматически после перезагрузки или перелогина. Вместо перезагрузки можно выполнить команду `source ~/.profile`, что вызовет принудительное обновление PATH.

```

ansible@ansible:~$ source ~/.profile
ansible@ansible:~$

```

- Теперь убедитесь, что ansible установился: `ansible --version`.

В результате вы увидите довольно подробный вывод о самой версии ansible, о путях к конфигам, версии python и т. д.

```

ansible@ansible:~$ ansible --version
ansible [core 2.16.6]
  config file = None
  configured module search path = ['/home/ansible/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /home/ansible/.local/lib/python3.10/site-packages/ansible
  ansible collection location = /home/ansible/.ansible/collections:/usr/share/ansible/collections
  executable location = /home/ansible/.local/bin/ansible
  python version = 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] (/usr/bin/python3)
  Jinja2 version = 3.0.3
  libyaml = True
ansible@ansible:~$

```

Машина с Docker Compose

- Для удобства дальнейшей работы, подключитесь к машине по ssh например при помощи KiTTY.

```
internet@internet: ~  
Файл  Правка  Вид  Поиск  Терминал  Вкладки  Справка  
ansible@ansible: ~  x  internet@internet: ~  x  +  ▾  
kemran@kemran:~$ ssh internet@192.168.0.107  
The authenticity of host '192.168.0.107 (192.168.0.107)' can't be established.  
ED25519 key fingerprint is SHA256:v+2y2R0ck2fognMDCxNr0tIoesGPug9H1AE8S7KkHq4.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '192.168.0.107' (ED25519) to the list of known hosts.  
internet@192.168.0.107's password:  
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-107-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/pro  
  
System information as of Thu May 16 12:14:54 PM UTC 2024  
  
System load:  0.04          Processes:            110  
Usage of /:   21.1% of 23.45GB  Users logged in:     1  
Memory usage: 12%          IPv4 address for enp0s3: 192.168.0.107  
Swap usage:   0%  
  
Expanded Security Maintenance for Applications is not enabled.  
  
3 updates can be applied immediately.  
To see these additional updates run: apt list --upgradable  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings  
  
Last login: Thu May 16 12:11:21 2024  
internet@internet:~$
```

2. Добавьте в список dns-серверов гугловский (8.8.8.8) и(или) яндексовый (77.88.8.8).

```
GNU nano 6.2 /etc/resolv.conf
# This is /run/systemd/resolve/stub-resolv.conf managed by man:systemd-resolved(
# Do not edit.
#
# This file might be symlinked as /etc/resolv.conf. If you're looking at
# /etc/resolv.conf and seeing this text, you have followed the symlink.
#
# This is a dynamic resolv.conf file for connecting local clients to the
# internal DNS stub resolver of systemd-resolved. This file lists all
# configured search domains.
#
# Run "resolvectl status" to see details about the uplink DNS servers
# currently in use.
#
# Third party programs should typically not access this file directly, but only
# through the symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a
# different way, replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes of
# operation for /etc/resolv.conf.
nameserver 8.8.8.8
nameserver 77.88.8.8
nameserver 127.0.0.53
options edns0 trust-ad
search .
```

3. Обновите индексы пакетов: `sudo apt-get update`.

```
internet@internet:~$ sudo apt-get update
Hit:1 http://ru.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://ru.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Hit:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:4 http://ru.archive.ubuntu.com/ubuntu jammy-backports InRelease
Fetched 119 kB in 1s (131 kB/s)
Reading package lists... Done
internet@internet:~$
```

4. Устанавливать docker и docker compose будем из официальных репозиториях Docker (копируйте команды построчно, до конечного \, если он есть):


```

internet@internet:~$ sudo install -m 0755 -d /etc/apt/keyrings
internet@internet:~$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
internet@internet:~$ sudo chmod a+r /etc/apt/keyrings/docker.asc
internet@internet:~$ echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
internet@internet:~$ sudo apt-get update
Hit:1 http://ru.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ru.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:4 http://ru.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:5 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Get:6 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [32.0 kB]
Fetched 80.9 kB in 1s (91.8 kB/s)
Reading package lists... Done
internet@internet:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done

```

```

internet@internet:~$ sudo usermod -aG docker $USER
internet@internet:~$ docker version
Client: Docker Engine - Community
Version: 26.1.2
API version: 1.45
Go version: go1.21.10
Git commit: 211e74b
Built: Wed May 8 13:59:59 2024
OS/Arch: linux/amd64
Context: default
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Frun%2Fdocker.sock/v1.45/version": dial unix /var/run/docker.sock: connect: permission denied
internet@internet:~$ docker compose version
Docker Compose version v2.27.0
internet@internet:~$ sudo usermod -aG docker internet
internet@internet:~$ docker version
Client: Docker Engine - Community
Version: 26.1.2
API version: 1.45
Go version: go1.21.10
Git commit: 211e74b
Built: Wed May 8 13:59:59 2024
OS/Arch: linux/amd64
Context: default
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Frun%2Fdocker.sock/v1.45/version": dial unix /var/run/docker.sock: connect: permission denied
internet@internet:~$

```

Установим некоторые нужные пакеты:

```
sudo apt-get install ca-certificates curl
```

Добавим официальный ключ для доступа к репозиторию Docker:

```

sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

```

Следующая команда настроит репозиторий:

```

echo \
"deb [arch=$(dpkg --print-architecture) \
signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

```

После добавления репозитория обновим индексы пакетов:

```
sudo apt-get update
```

Теперь установим docker и docker compose:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin  
docker-compose-plugin
```

Добавьте текущего пользователя в группу docker:

```
sudo usermod -aG docker $USER
```

- Перезагрузите сервер или перелогиньтесь;
- Проверьте, что установка прошла успешно:

```
docker compose version
```

В данном случае мы установили плагин "compose" для docker, но есть и отдельный пакет "docker-compose" (как видно разница в наличии -). На данный момент он считается устаревшим.

Создаём сервера

1. Выполните команду `ip a` и найдите сетевой интерфейс, который подключён к вашему роутеру.

Здесь же можно определить адрес подсети. После `ip`-адреса машины указано `/24` - это маска подсети. Она говорит о том, что первые 24 бита `ip`-адреса фиксированы (т.е. первые 3 числа), а оставшиеся 8 бит (т.е. последнее число) могут изменяться. Очевидно что в такой подсети есть всего 256 различных `ip`-адресов, два из которых служебные: `192.168.1.255` - занят под broadcast (для отправки пакета всем в подсети) и `192.168.1.0` - который и называется **адресом подсети**.

```
internet@internet:~$ 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 1000
    link/ether 08:00:27:b4:29:e5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.106/24 metric 100 brd 192.168.0.255 scope global dynamic enp0s3
        valid_lft 6103sec preferred_lft 6103sec
    inet6 fe80::a00:27ff:feb4:29e5/64 scope link
        valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:23:c8:ef:90 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
internet@internet:~$
```

2. Выполните команду `ip r` и найдите `default`. Этот `ip` - **gateway** (в нашем случае это `ip` роутера).

```
internet@internet:~$ ip r
default via 192.168.0.1 dev enp0s3 proto dhcp src 192.168.0.106 metric 100
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.0.0/24 dev enp0s3 proto kernel scope link src 192.168.0.106 metric 100
192.168.0.1 dev enp0s3 proto dhcp scope link src 192.168.0.106 metric 100
internet@internet:~$
```

3. Теперь нужно посмотреть занятые ip адреса в нашей локальной сети. Это нужно, чтобы выбрать диапазон свободных для наших серверов.
Установите утилиту `arp` при помощи команды: `sudo apt install net-tools`, а затем выполните `arp -e`. IP-адреса из списка заняты.

Список скорее всего будет не полным, т.к. он содержит только те IP к которым обращалась ваша машина или которые обращались к ней.

```
internet@internet:~$ sudo apt install net-tools
[sudo] password for internet:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  net-tools
0 upgraded, 1 newly installed, 0 to remove and 4 not upgraded.
Need to get 204 kB of archives.
After this operation, 819 kB of additional disk space will be used.
Get:1 http://ru.archive.ubuntu.com/ubuntu jammy/main amd64 net-tools amd64 1.60+git20181103.0eebece-1ubu
Fetched 204 kB in 1s (207 kB/s)
Selecting previously unselected package net-tools.
(Reading database ... 74815 files and directories currently installed.)
Preparing to unpack .../net-tools_1.60+git20181103.0eebece-1ubuntu5_amd64.deb ...
Unpacking net-tools (1.60+git20181103.0eebece-1ubuntu5) ...
Setting up net-tools (1.60+git20181103.0eebece-1ubuntu5) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
internet@internet:~$ arp -e
Address      HWtype  HWaddress      Flags Mask      Iface
_gateway     ether    48:22:54:8f:ad:38  C               enp0s3
192.168.0.105 ether    b8:1e:a4:74:eb:dd  C               enp0s3
internet@internet:~$
```

4. Чтобы уточнить список воспользуемся утилитой `nmap`. Установите её (`sudo apt -get install nmap`) и попросите просканировать все ip в сети (`sudo nmap -Pn 192.168.1.0/24`). К сожалению, при помощи `nmap` тоже не всегда можно получить полный список занятых адресов, т.к. некоторые устройства могут временно отключаться от сети для экономии батарее и т.д.
Точный способ узнать занятые IP - посмотреть их в web-интерфейсе роутера.

```

internet@internet:~$ sudo nmap -Pn 192.168.0.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2024-05-16 13:03 UTC
Nmap scan report for _gateway (192.168.0.1)
Host is up (0.0046s latency).
Not shown: 996 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
80/tcp    open  http
1900/tcp  open  upnp
MAC Address: 48:22:54:8F:AD:38 (Unknown)

Nmap scan report for 192.168.0.100
Host is up (0.0043s latency).
All 1000 scanned ports on 192.168.0.100 are closed
MAC Address: 12:FA:14:65:83:B5 (Unknown)

Nmap scan report for 192.168.0.101
Host is up (0.011s latency).
All 1000 scanned ports on 192.168.0.101 are closed
MAC Address: 4A:AC:FC:D1:7A:9B (Unknown)

Nmap scan report for 192.168.0.102
Host is up (0.0041s latency).
All 1000 scanned ports on 192.168.0.102 are filtered
MAC Address: 4C:BC:E9:1E:B0:D2 (Unknown)

Nmap scan report for 192.168.0.105
Host is up (0.00018s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: B8:1E:A4:74:EB:DD (Unknown)

Nmap scan report for 192.168.0.107
Host is up (0.00049s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: 08:00:27:9E:9F:FC (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.0.108
Host is up (0.0099s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
1034/tcp  open  zincite-a
3000/tcp  open  ppp
3001/tcp  open  nessus
9998/tcp  open  distinct32
MAC Address: 2C:2B:F9:61:C9:C6 (LG Innotek)

Nmap scan report for internet (192.168.0.106)
Host is up (0.000040s latency).

```

Теперь воспользуемся docker compose чтобы создать сервера. Для начала пропишем в нём только один сервер.

1. На сервере с docker compose создайте файл `compose.yaml` и откройте его в текстовом редакторе.
2. Поместите в него следующий текст (с учётом ваших параметров сети):

```

services:
  test: # Так я назвал сервис
    image: rastasheep/ubuntu-sshd # Тот самый базовый образ
    dns:
      - 77.88.8.8 # Яндексский
      - 8.8.8.8   # Гугловый

```

```

networks:
  outside:
    ipv4_address: 192.168.1.100 # Свободный ip из нашей подсети

# Здесь создаём docker-сеть
networks:
  outside: # Так я назвал сеть
    driver: ipvlan
    driver_opts:
      parent: enp0s3 # Сетевой интерфейс
    ipam:
      config:
        - subnet: 192.168.1.0/24 # Адрес подсети и маска
          gateway: 192.168.1.1

```

Как видно по описанию создаётся одна машина и одна сеть.

```

GNU nano 6.2                                compose.yaml
services:
  test: # Так я назвал сервис
    image: rastasheep/ubuntu-sshd # Тот самый базовый образ
    dns:
      - 77.88.8.8 # Яндексский
      - 8.8.8.8   # Гугловый
    networks:
      outside:
        ipv4_address: 192.168.0.99 # Свободный ip из нашей подсети

# Здесь создаём docker-сеть
networks:
  outside: # Так я назвал сеть
    driver: ipvlan
    driver_opts:
      parent: enp0s3 # Сетевой интерфейс
    ipam:
      config:
        - subnet: 192.168.0.0/24 # Адрес подсети и маска
          gateway: 192.168.0.1

```

3. Наберите в терминале:
docker compose up -d

В результате, docker compose найдёт файл `compose.yaml` в текущей папке, создаст сеть `outside` и запустит машину `test` в фоновом режиме (`-d`), т.е не захватывая терминал и не выводя логи.

```
internet@internet:~/compose$ docker compose up -d
[+] Running 14/14
✓ test Pulled
✓ a48c500ed24e Pull complete
✓ 1e1de00ff7e1 Pull complete
✓ 0330ca45a200 Pull complete
✓ 471db38bcfbf Pull complete
✓ 0b4aba487617 Pull complete
✓ b42109ad2a3d Pull complete
✓ dde737735b18 Pull complete
✓ d836c14266f7 Pull complete
✓ 5ed86b5d4a15 Pull complete
✓ 5273c120f396 Pull complete
✓ b0299e0551df Pull complete
✓ 0ae38e059780 Pull complete
✓ ca79c723275f Pull complete
[+] Running 2/2
✓ Network compose_outside Created
✓ Container compose-test-1 Started
internet@internet:~/compose$
```

4. Проверьте, что к этой машине можно подключиться как и к любой другой, например при помощи KiTTY (ip, port и пользователь вам известны).

```
kemran@kemran:~$ ssh root@192.168.0.99
root@192.168.0.99's password:
Last login: Thu May 16 15:39:34 2024 from 192.168.0.105
root@d2db4a2c0992:~#
```

5. Наберите в терминале:

docker compose down

В результате, docker compose остановит машину и удалит созданные им контейнер и сеть.

Внимание: после остановки машины будут уничтожены и все изменения, которые в них были сделаны тоже исчезнут. Если хотите приостановить машины на время, без удаления, используйте `docker compose stop` и для последующего возобновления работы `docker compose start`. После остановки все запущенные на машине процессы тоже будут остановлены, но файлы и установленный софт останется.

```
internet@internet:~/compose$ docker compose down
[+] Running 2/1
✓ Container compose-test-1 Removed
✓ Network compose_outside Removed
internet@internet:~/compose$
```

Модифицируем файл таким образом, чтобы создать 3 машины: одну под - балансировщик нагрузки, вторую под базу данных и третью под приложение (пока одну, затем их будет больше).

1. Откройте файл `compose.yaml` и поместите следующий текст (с учётом ваших параметров сети):
services:

```
load_balancer: # Балансировщик нагрузки
  image: rastasheep/ubuntu-sshd
  dns:
    - 77.88.8.8
    - 8.8.8.8
  networks:
    outside:
      ipv4_address: 192.168.1.100
```

```
db: # База данных
  image: rastasheep/ubuntu-sshd
  dns:
    - 77.88.8.8
    - 8.8.8.8
  networks:
    outside:
      ipv4_address: 192.168.1.200
```

```
worker: # Сервер с приложением
  image: rastasheep/ubuntu-sshd
  dns:
    - 77.88.8.8
    - 8.8.8.8
  networks:
    outside:
      ipv4_address: 192.168.1.101
```

```
networks:
  outside:
    driver: ipvlan
    driver_opts:
      parent: enp0s3
    ipam:
      config:
        - subnet: 192.168.1.0/24
          gateway: 192.168.1.1
```

Запомните ip-адреса серверов, они будут нужны ansible.

```

GNU nano 6.2                                compose.yaml *
services:
  test: # Так я назвал сервис
    image: rastasheep/ubuntu-sshd # Тот самый базовый образ
    dns:
      - 77.88.8.8 # Яндексский
      - 8.8.8.8   # Гугловый
    networks:
      outside:
        ipv4_address: 192.168.0.99 # Свободный ip из нашей подсети

  db: # База данных
    image: rastasheep/ubuntu-sshd
    dns:
      - 77.88.8.8
      - 8.8.8.8
    networks:
      outside:
        ipv4_address: 192.168.0.98

  worker: # Сервер с приложением
    image: rastasheep/ubuntu-sshd
    dns:
      - 77.88.8.8
      - 8.8.8.8
    networks:
      outside:
        ipv4_address: 192.168.0.97

# Здесь создаём docker-сеть
networks:
  outside: # Так я назвал сеть
    driver: ipvlan
    driver_opts:
      parent: enp0s3 # Сетевой интерфейс
    ipam:
      config:
        - subnet: 192.168.0.0/24 # Адрес подсети и маска
          gateway: 192.168.0.1

```

2. Запустите машины командой: `docker compose up -d` и на некоторое время мы перейдём на машину с ansible.

```

internet@internet:~/compose$ docker compose up -d
[+] Running 4/4
✔ Network compose_outside      Created                                0.0s
✔ Container compose-db-1       Started                             0.5s
✔ Container compose-worker-1   Started                             0.4s
✔ Container compose-load_balancer-1 Started                             0.6s
internet@internet:~/compose$

```

Соединяем Ansible с управляемыми машинами

1. Выполните команду:


```
ansible@ansible:~$ ansible-inventory --list
[WARNING]: No inventory was parsed, only implicit localhost is available
{
  "_meta": {
    "hostvars": {}
  },
  "all": {
    "children": [
      "ungrouped"
    ]
  }
}
```

2. Выполните команду:

```
ansible@ansible:~$ ansible --version | grep config
config file = None
configured module search path = ['/home/ansible/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
ansible@ansible:~$
```

3. Создайте каталог "app" и в нём "ansible", затем перейдите туда. Это не какие-то специальные названия, просто чтобы была понятная структура.

```
ansible@ansible:~$ mkdir app
ansible@ansible:~$ mkdir app/ansible
ansible@ansible:~$ cd app/ansible/
ansible@ansible:~/app/ansible$
```

4. Создайте файл "ansible.cfg" содержащий:

```
GNU nano 6.2                                     ansible.cfg
[defaults]
inventory = ./hosts
```

5. Снова выполните команду:

```
ansible@ansible:~/app/ansible$ ansible --version | grep config
config file = /home/ansible/app/ansible/ansible.cfg
configured module search path = ['/home/ansible/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
ansible@ansible:~/app/ansible$ ~
```

6. В конфиге мы указали, что файл инвентаря лежит в той же папке, что и конфиг и называется "hosts". Обычно так и поступают, но технически можно было положить инвентарь в другое место.

```
GNU nano 6.2                                hosts
load_balancer ansible_host=192.168.0.99 ansible_user=root ansible_password=root
db ansible_host=192.168.0.98 ansible_user=root ansible_password=root
worker ansible_host=192.168.0.97 ansible_user=root ansible_password=root
```

7. Создайте этот файл и напишите в нём следующее (только ip-адреса укажите свои):

```
GNU nano 6.2                                hosts
load_balancer ansible_host=192.168.0.99 ansible_user=root ansible_password=root
db ansible_host=192.168.0.98 ansible_user=root ansible_password=root
worker ansible_host=192.168.0.97 ansible_user=root ansible_password=root
```

8. Убедитесь, что ansible подхватил информацию о хостах: `ansible-inventory -list`.

```

ansible@ansible:~/app/ansible$ ansible-inventory --list
{
  "_meta": {
    "hostvars": {
      "db": {
        "ansible_host": "192.168.0.98",
        "ansible_password": "root",
        "ansible_user": "root"
      },
      "load_balancer": {
        "ansible_host": "192.168.0.99",
        "ansible_password": "root",
        "ansible_user": "root"
      },
      "worker": {
        "ansible_host": "192.168.0.97",
        "ansible_password": "root",
        "ansible_user": "root"
      }
    }
  },
  "all": {
    "children": [
      "ungrouped"
    ]
  },
  "ungrouped": {
    "hosts": [
      "load_balancer",
      "db",
      "worker"
    ]
  }
}
ansible@ansible:~/app/ansible$

```

9. Попробуем подключиться к ним.

Ansible может выполнять команды в двух режимах: одиночная команда (ad hoc) или сценарий (play). Общий вид запуска одиночной команды выглядит так:

```
ansible all -m ping
```

```

ansible@ansible:~/app/ansible$ ansible all -m ping
load_balancer | FAILED! => {
  "msg": "to use the 'ssh' connection type with passwords or pkcs11_provider, you must install the sshpass program"
}
db | FAILED! => {
  "msg": "to use the 'ssh' connection type with passwords or pkcs11_provider, you must install the sshpass program"
}
worker | FAILED! => {
  "msg": "to use the 'ssh' connection type with passwords or pkcs11_provider, you must install the sshpass program"
}
ansible@ansible:~/app/ansible$

```

10. В итоге вы должны были получить ошибку, т.к. для авторизации по паролю нужен дополнительный пакет: `sshpass`. Установите его при помощи `apt` и повторите `ping`.

```

ansible@ansible:~/app/ansible$ sudo apt install sshpass
[sudo] password for ansible:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  sshpass
0 upgraded, 1 newly installed, 0 to remove and 2 not upgraded.
Need to get 11.7 kB of archives.
After this operation, 35.8 kB of additional disk space will be used.
Get:1 http://ru.archive.ubuntu.com/ubuntu jammy/universe amd64 sshpass amd64 1.09-1 [11.7 kB]
Fetched 11.7 kB in 0s (30.9 kB/s)
Selecting previously unselected package sshpass.
(Reading database ... 81348 files and directories currently installed.)
Preparing to unpack .../sshpass_1.09-1_amd64.deb ...
Unpacking sshpass (1.09-1) ...
Setting up sshpass (1.09-1) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ansible@ansible:~/app/ansible$

```

11. В этот раз текст ошибки должен измениться. Суть ошибки в том, что мы не подтвердили, что доверяем этим серверам (в работе №2 мы отвечали yes при первом подключении). Для решения этой проблемы есть 2 основных способа:

```

ansible@ansible:~/app/ansible$ ansible all -m ping
worker | FAILED! => {
  "msg": "Using a SSH password instead of a key is not possible because Host Key checking is enabled and sshpass does not support this. Please add this host's fingerprint to your known_hosts file to manage this host."
}
db | FAILED! => {
  "msg": "Using a SSH password instead of a key is not possible because Host Key checking is enabled and sshpass does not support this. Please add this host's fingerprint to your known_hosts file to manage this host."
}
load_balancer | FAILED! => {
  "msg": "Using a SSH password instead of a key is not possible because Host Key checking is enabled and sshpass does not support this. Please add this host's fingerprint to your known_hosts file to manage this host."
}
ansible@ansible:~/app/ansible$

```

Приложения

×

Файл Правка Вид Поиск Терминал

ansible@ansible: ~/app/ansible

GNU nano 6.2

[defaults]

inventory = ./hosts

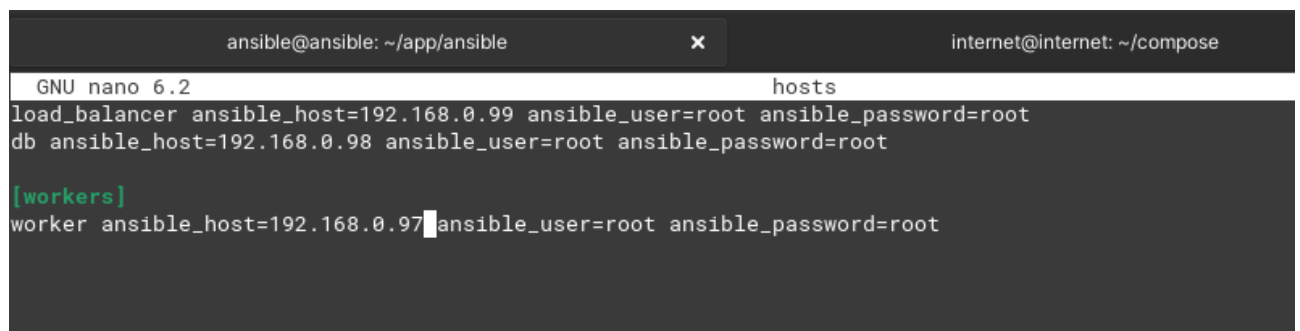
host_key_checking = false

12. В этот раз подключение должно быть успешно, и все три сервера должны ответить "pong".

```
ansible@ansible: ~/app/ansible$ ansible all -m ping
worker | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
load_balancer | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
db | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
ansible@ansible: ~/app/ansible$
```

Разделение на группы

1. Откройте файл "hosts" и замените его на (со своими ip):



```
ansible@ansible: ~/app/ansible x internet@internet: ~/compose
GNU nano 6.2 hosts
load_balancer ansible_host=192.168.0.99 ansible_user=root ansible_password=root
db ansible_host=192.168.0.98 ansible_user=root ansible_password=root

[workers]
worker ansible_host=192.168.0.97 ansible_user=root ansible_password=root
```

2. Перейдите на виртуальную машину с docker compose, остановите запущенные машины и добавьте в "compose.yaml" ещё 3 машины. В моём случае это будет выглядеть так:

```

GNU nano 6.2                                compose.yaml
services:
  load_balancer: # Балансировщик нагрузки
    image: rastasheep/ubuntu-sshd
    dns:
      - 77.88.8.8
      - 8.8.8.8
    networks:
      outside:
        ipv4_address: 192.168.0.99

  db: # База данных
    image: rastasheep/ubuntu-sshd
    dns:
      - 77.88.8.8
      - 8.8.8.8
    networks:
      outside:
        ipv4_address: 192.168.0.99

  worker1: # Сервер с приложением
    image: rastasheep/ubuntu-sshd
    dns:
      - 77.88.8.8
      - 8.8.8.8
    networks:
      outside:
        ipv4_address: 192.168.0.98

  worker2: # Сервер с приложением
    image: rastasheep/ubuntu-sshd
    dns:
      - 77.88.8.8
      - 8.8.8.8
    networks:
      outside:
        ipv4_address: 192.168.0.97

  worker3: # Сервер с приложением
    image: rastasheep/ubuntu-sshd
    dns:
      - 77.88.8.8
      - 8.8.8.8
    networks:
      outside:
        ipv4_address: 192.168.0.96

  worker4: # Сервер с приложением
    image: rastasheep/ubuntu-sshd
    dns:
      - 77.88.8.8
      - 8.8.8.8

```

3. Запустите машины и вернитесь на сервер с ansible.

```

internet@internet:~/compose$ docker compose up -d
WARN[0000] Found orphan containers ([compose-worker-1]) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up.
[*] Running ???
✓ Network compose_outside          Created
✓ Container compose-worker2-1      Started
✓ Container compose-worker3-1      Started
✓ Container compose-worker4-1      Started
✓ Container compose-load_balancer-1 Started
✓ Container compose-db-1           Started
✓ Container compose-worker1-1      Started
internet@internet:~/compose$

```

4. Т.к. у нас появилось несколько новых машин, нам нужно добавить их в инвентарь.

Снова откройте файл "hosts" и замените его на (со своими ip):

```

GNU nano 6.2                                     hosts
load_balancer ansible_host=192.168.0.99 ansible_user=root ansible_password=root
db ansible_host=192.168.0.98 ansible_user=root ansible_password=root

[workers]
worker1 ansible_host=192.168.0.97 ansible_user=root ansible_password=root
worker2 ansible_host=192.168.0.96 ansible_user=root ansible_password=root
worker3 ansible_host=192.168.0.95 ansible_user=root ansible_password=root

[staging]
worker4 ansible_host=192.168.0.94 ansible_user=root ansible_password=root

[workers:children]
prod
staging

```

5. Проверьте, что машины доступны при помощи модуля `ansible ping`.
Прошпигуйте все воркеры и только те, которые находятся в группе "prod".

```

GNU nano 6.2                                     hosts
load_balancer ansible_host=192.168.0.99 ansible_user=root ansible_password=root
db ansible_host=192.168.0.98 ansible_user=root ansible_password=root

[workers]
worker1 ansible_host=192.168.0.97 ansible_user=root ansible_password=root
worker2 ansible_host=192.168.0.96 ansible_user=root ansible_password=root
worker3 ansible_host=192.168.0.95 ansible_user=root ansible_password=root

[staging]
worker4 ansible_host=192.168.0.94 ansible_user=root ansible_password=root

[workers:children]
prod
staging

```

Изменяем способ доступа к серверам

1. Выполните команду:

```

ansible@ansible:~/app/ansible$ cp hosts hosts_root
ansible@ansible:~/app/ansible$

```

2. Проверьте, что копирование файла прошло успешно. Для этого пропингуйте все сервера командой:

```

ansible@ansible:~/app/ansible$ ansible -i hosts_root all -m ping
db | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
worker3 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
worker1 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
worker2 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
load_balancer | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
worker4 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
ansible@ansible:~/app/ansible$

```

3. Обновим индексы apt-пакетов на сервере, для этого будем использовать модуль ansible [apt](#):

```

ansible@ansible:~/app/ansible$ ansible staging -m apt -a "update_cache=yes"
worker4 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1715892134,
  "cache_updated": false,
  "changed": false
}
ansible@ansible:~/app/ansible$

```


4. Установим пакет `sudo` (да, это отдельный пакет, а не встроенная команда):

```
ansible@ansible:~/app/ansible$ ansible staging -m apt -a "name=sudo state=latest"
worker4 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "cache_update_time": 1715892134,
  "cache_updated": false,
  "changed": true,
  "stderr": "debconf: delaying package configuration, since apt-utils is not installed\n",
  "stderr_lines": [
    "debconf: delaying package configuration, since apt-utils is not installed"
  ],
  "stdout": "Reading package lists...\nBuilding dependency tree...\nReading state information...\nThe following NEW
packages will be installed:\n  sudo\n0 upgraded, 1 newly installed, 0 to remove and 122 not upgraded.\nNeed to get 430
kB of archives.\nAfter this operation, 1765 kB of additional disk space will be used.\nGet:1 http://archive.ubuntu.co
m/ubuntu bionic-updates/main amd64 sudo amd64 1.8.21p2-3ubuntu1.6 [430 kB]\nFetched 430 kB in 1s (704 kB/s)\nSelecting
previously unselected package sudo.\n\n(Reading database ... 5%\r(Reading database ... 10%\r(
Reading database ... 15%\r(Reading database ... 20%\r(Reading database ... 25%\r(Reading database ... 30%\r(Reading da
tabase ... 35%\r(Reading database ... 40%\r(Reading database ... 45%\r(Reading database ... 50%\r(Reading database ...
55%\r(Reading database ... 60%\r(Reading database ... 65%\r(Reading database ... 70%\r(Reading database ... 75%\r(Rea
ding database ... 80%\r(Reading database ... 85%\r(Reading database ... 90%\r(Reading database ... 95%\r(Reading datab
ase ... 100%\r(Reading database ... 10011 files and directories currently installed.)\r\nPreparing to unpack .../sudo_
1.8.21p2-3ubuntu1.6_amd64.deb ... \r\nUnpacking sudo (1.8.21p2-3ubuntu1.6) ... \r\nSetting up sudo (1.8.21p2-3ubuntu1.6)
... \r\n",
  "stdout_lines": [
    "Reading package lists...",
    "Building dependency tree...",
    "Reading state information...",
    "The following NEW packages will be installed:",
    " sudo",
    "0 upgraded, 1 newly installed, 0 to remove and 122 not upgraded.",
    "Need to get 430 kB of archives.",
    "After this operation, 1765 kB of additional disk space will be used.",
    "Get:1 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 sudo amd64 1.8.21p2-3ubuntu1.6 [430 kB]",
    "Fetched 430 kB in 1s (704 kB/s)",
    "Selecting previously unselected package sudo.",
    "(Reading database ... ",
    "(Reading database ... 5%",
    "(Reading database ... 10%",
    "(Reading database ... 15%",
    "(Reading database ... 20%",
    "(Reading database ... 25%",
    "(Reading database ... 30%",
    "(Reading database ... 35%",
    "(Reading database ... 40%",
    "(Reading database ... 45%",
    "(Reading database ... 50%",
    "(Reading database ... 55%",
    "(Reading database ... 60%",
    "(Reading database ... 65%",
    "(Reading database ... 70%",
    "(Reading database ... 75%",
```

5. По умолчанию новые пользователи обладают минимальными правами, чтобы дать возможность пользователю повышать свои привилегии при помощи `sudo` он или группа в которой он состоит должны быть записаны в файле `/etc/sudoers` с указанием доступных привилегий. Создадим новую группу с названием `"ansible"` при помощи модуля [group](#):

```

ansible@ansible:~/app/ansible$ ansible staging -m group -a'name=ansible state=present'
worker4 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": true,
  "gid": 1000,
  "name": "ansible",
  "state": "present",
  "system": false
}
ansible@ansible:~/app/ansible$

```

- Разрешим пользователям из этой группы выполнять любые команды, при этом без необходимости вводить пароль (нужно для ansible).

Чтобы случайно не поломать файл `/etc/sudoers` внося в него изменения, мы создадим новый файл в каталоге `/etc/sudoers.d/` (файл `/etc/sudoers` подтягивает все файлы из этого каталога):

```

ansible@ansible:~/app/ansible$ ansible staging -m copy -a"content='%ansible ALL=(ALL:ALL) NOPASSWD:ALL' dest=/etc/sudoers.d/ansible validate='/usr/sbin/visudo -cf %s'"
worker4 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": true,
  "checksum": "18a81547f64e8f74b52c8871ba04227641ab2d06",
  "dest": "/etc/sudoers.d/ansible",
  "gid": 0,
  "group": "root",
  "md5sum": "2d654c6d64d6f3d8117d30adc5158e45",
  "mode": "0644",
  "owner": "root",
  "size": 35,
  "src": "/root/.ansible/tmp/ansible-tmp-1715892231.1562583-1746-128769534761639/source",
  "state": "file",
  "uid": 0
}
ansible@ansible:~/app/ansible$

```

- Теперь настало время создать пользователя под которым в дальнейшем будет работать ansible:

```

ansible@ansible:~/app/ansible$ ansible staging -m user -a"name=ansible shell=/bin/bash groups=ansible append=yes password={{ '123' | password_hash('sha512') }}" update_password=on_create"
[DEPRECATION WARNING]: Encryption using the Python crypt module is deprecated. The Python crypt module is deprecated and will be removed from Python 3.13. Install the passlib library for continued encryption functionality. This feature will be removed in version 2.17. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
worker4 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": true,
  "comment": "",
  "create_home": true,
  "group": 100,
  "groups": "ansible",
  "home": "/home/ansible",
  "name": "ansible",
  "password": "NOT_LOGGING_PASSWORD",
  "shell": "/bin/bash",
  "state": "present",
  "system": false,
  "uid": 1000
}
ansible@ansible:~/app/ansible$

```

8. Настроим на машинах из группы "staging" доступ под пользователем "ansible" по ключу.

Для этого первым делом нам нужна пара приватный/публичный ключ которую мы будем использовать для доступа. В принципе на нашем сервере уже есть такая пара (она была создана при установке ssh-сервера), но мы создадим ещё одну.

```
ansible@ansible:~/app/ansible$ mkdir keys
ansible@ansible:~/app/ansible$ cd keys/
ansible@ansible:~/app/ansible/keys$ ssh-keygen -t rsa -b 4096 -f ansible_key
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ansible_key
Your public key has been saved in ansible_key.pub
The key fingerprint is:
SHA256:S/BJB8xYBnQuPKWjdSYH97dqvpXk99N4ocEQ4AlwQFw ansible@ansible
The key's randomart image is:
+---[RSA 4096]-----+
|  o==0@o*          |
|  .oB=0.o          |
|  = +O. o .        |
|  . E o . . .      |
|  o S   o o        |
|  o .   * o        |
|  .   o *.+        |
|  o o.o+          |
|  o. .o           |
+----[SHA256]-----+
ansible@ansible:~/app/ansible/keys$ ansible staging -m authorized_key -a"user=ansible key=\"{{ lookup('file', '/home/' + lookup('env', 'USER') + '/app/ansible/keys/ansible_key.pub') }}\""
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'
[WARNING]: Could not match supplied host pattern, ignoring: staging
ansible@ansible:~/app/ansible/keys$
```

9. Отключим пользователю "root" пароль, чтобы ни у кого не было возможности залогиниться под ним:

```
ansible@ansible:~/app/ansible$ ansible staging -m user -a"name=root password='*' update_password=always"
worker4 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "append": false,
  "changed": true,
  "comment": "root",
  "group": 0,
  "home": "/root",
  "move_home": false,
  "name": "root",
  "password": "NOT_LOGGING_PASSWORD",
  "shell": "/bin/bash",
  "state": "present",
  "uid": 0
}
```

10. Подождите несколько минут и запустите пинг **всех** серверов (через ansible). Вы должны увидеть, что сервер из группы "staging" не ответит.

```

ansible@ansible:~/app/ansible$ ansible -i hosts_root all -m ping
worker1 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
db | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
load_balancer | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
worker2 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
worker3 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
worker4 | UNREACHABLE! => {
  "changed": false,
  "msg": "Invalid/incorrect password: Permission denied, please try again.",
  "unreachable": true
}
ansible@ansible:~/app/ansible$

```

11. Внесём изменения в файл "hosts", чтобы наладить связь с сервером.

Откройте "hosts" в текстовом редакторе и для "worker4" (т.к. он один у нас в группе "staging") замените `ansible_user=root` на `ansible_user=ansible` и `ansible_password=root` на `ansible_ssh_private_key_file=keys/ansible_key`.

Здесь путь к приватному ключу указан относительно расположения файла "hosts".

```

GNU nano 6.2                                hosts
load_balancer ansible_host=192.168.0.99 ansible_user=root ansible_password=root
db ansible_host=192.168.0.98 ansible_user=root ansible_password=root

[prod]
worker1 ansible_host=192.168.0.97 ansible_user=root ansible_password=root
worker2 ansible_host=192.168.0.96 ansible_user=root ansible_password=root
worker3 ansible_host=192.168.0.95 ansible_user=root ansible_password=root

[staging]
worker4 ansible_host=192.168.0.94 ansible_user=ansible ansible_ssh_private_key_file=keys/ansible_key

[workers:children]
prod
staging

```

12. Убедитесь, что теперь все сервера отвечают на ping.

```

ansible@ansible:~/app/ansible$ ansible all -m ping
worker3 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
load_balancer | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
worker1 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
db | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
worker2 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
worker4 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
ansible@ansible:~/app/ansible$

```

Пишем сценарий

1. Создайте в каталоге "~/app/ansible" файл "ping.yaml";
2. Добавьте в файл следующие строки:

```
GNU nano 6.2                                ping.yaml
- name: Ping all servers
  hosts: all
```

3. Теперь добавим непосредственно команды:

```
GNU nano 6.2                                ping.yaml *
```

```
- name: Ping all servers
  hosts: all
  tasks:
    - name: Ping all servers
      ping:
```

4. Запустите playbook при помощи команды:

```
ansible@ansible:~/app/ansible$ ansible-playbook ping.yaml

PLAY [Ping all servers] *****

TASK [Gathering Facts] *****
ok: [load_balancer]
ok: [worker2]
ok: [worker3]
ok: [db]
ok: [worker1]
ok: [worker4]

TASK [Ping all servers] *****
ok: [worker3]
ok: [load_balancer]
ok: [worker2]
ok: [worker1]
ok: [db]
ok: [worker4]

PLAY RECAP *****
db                                : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
load_balancer                    : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
worker1                          : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
worker2                          : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
worker3                          : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
worker4                          : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ansible@ansible:~/app/ansible$
```

Теперь напишем playbook который будет делать ровно тоже, что мы делали руками в предыдущем разделе:

1. Создайте в каталоге "~/app/ansible" файл "setup.yaml";
2. Добавьте в файл следующие строки:
3. Теперь добавим непосредственно команды. Сравните ad hoc команды из предыдущего раздела с их вариантами в playbook:

```
GNU nano 6.2                                setup.yaml
name: Change access from root to ansible user and apt-package update servers
hosts: all
become: yes
tasks:
  - name: Change apt-package update servers
    shell: | # / в yml означает, что далее многострочный текст
            sed -i 's|archive.ubuntu.com|mirror.yandex.ru|g' /etc/apt/sources.list
            sed -i 's|security.ubuntu.com|mirror.yandex.ru|g' /etc/apt/sources.list
  - name: Install sudo
    apt:
      name: sudo
      state: latest
      update_cache: yes # Перед установкой делает update
  - name: Create group ansible
    group:
      name: ansible
      state: present
  - name: Add group to sudoers
    copy:
      content: '%ansible ALL=(ALL:ALL) NOPASSWD:ALL'
      dest: /etc/sudoers.d/ansible
      validate: '/usr/sbin/visudo -cf %s'
  - name: Create user ansible and add to ansible group
    user:
      name: ansible
      shell: /bin/bash
      groups: ansible
      append: yes
      password: "{{ '123' | password_hash('sha512') }}"
      update_password: on_create
  - name: Set ssh authorized key
    authorized_key:
      user: ansible
      key: "{{ lookup('file', '/home/' + lookup('env', 'USER') + '/app/ansible/keys/ansible_key.pub') }}"
  - name: Set empty root password
    user:
      name: root
      password: '*'
      update_password: always
```

4. Запустите playbook и дождитесь его завершения. В логах вы должны увидеть, что на всех серверах кроме "worker4" на каждом этапе происходили изменения (changed), а на "worker4" изменений не было (ok).


```

ansible@ansible:~/app/ansible$ ansible-playbook setup.yaml

PLAY [Change access from root to ansible user and apt-package update servers] *****

TASK [Gathering Facts] *****
ok: [db]
ok: [worker1]
ok: [worker2]
ok: [load_balancer]
ok: [worker3]
ok: [worker4]

TASK [Change apt-package update servers] *****
changed: [worker1]
changed: [db]
changed: [load_balancer]
changed: [worker2]
changed: [worker3]
changed: [worker4]

TASK [Install sudo] *****
changed: [worker1]
changed: [worker2]
changed: [load_balancer]
changed: [db]
changed: [worker3]
ok: [worker4]

TASK [Create group ansible] *****
changed: [load_balancer]
changed: [db]
changed: [worker3]
changed: [worker1]
changed: [worker2]
ok: [worker4]

TASK [Add group to sudoers] *****
changed: [worker3]
changed: [db]
changed: [load_balancer]
changed: [worker2]
changed: [worker1]
ok: [worker4]

TASK [Create user ansible and add to ansible group] *****
[DEPRECATION WARNING]: Encryption using the Python crypt module is deprecated. The Python crypt module is deprecated
and will be removed from Python 3.13. Install the passlib library for continued encryption functionality. This
feature will be removed in version 2.17. Deprecation warnings can be disabled by setting deprecation_warnings=False
in ansible.cfg.
changed: [worker2]
changed: [db]

```

5. Подождите несколько минут и запустите "ping.yaml". Теперь, все сервера, кроме "worker4" перестали отвечать, т.к. в hosts указаны данные для пользователя root.


```

ansible@ansible:~/app/ansible$ ansible-playbook ping.yaml

PLAY [Ping all servers] *****

TASK [Gathering Facts] *****
fatal: [db]: UNREACHABLE! => {"changed": false, "msg": "Invalid/incorrect password: Permission denied, please try again.", "unreachable": true}
fatal: [load_balancer]: UNREACHABLE! => {"changed": false, "msg": "Invalid/incorrect password: Permission denied, please try again.", "unreachable": true}
fatal: [worker2]: UNREACHABLE! => {"changed": false, "msg": "Invalid/incorrect password: Permission denied, please try again.", "unreachable": true}
fatal: [worker1]: UNREACHABLE! => {"changed": false, "msg": "Invalid/incorrect password: Permission denied, please try again.", "unreachable": true}
fatal: [worker3]: UNREACHABLE! => {"changed": false, "msg": "Invalid/incorrect password: Permission denied, please try again.", "unreachable": true}
ok: [worker4]

TASK [Ping all servers] *****
ok: [worker4]

PLAY RECAP *****
db                                : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ignored=0
load_balancer                    : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ignored=0
worker1                          : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ignored=0
worker2                          : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ignored=0
worker3                          : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ignored=0
worker4                          : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ansible@ansible:~/app/ansible$

```

6. Поправьте файл hosts и убедитесь, что теперь все сервера на связи.

```
ansible@ansible:~/app/ansible$ ansible all -m ping
load_balancer | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
worker2 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
db | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
worker3 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
worker1 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
worker4 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
ansible@ansible:~/app/ansible$
```

Установка софта на сервер базы данных

1. Создайте в каталоге "~/app/ansible" файл "install_mysql.yaml";
2. Добавьте в файл следующие строки:

```

GNU nano 6.2                                install_mysql.yaml
- name: Install_mysql
  hosts: db
  become: yes

  tasks:
    - name: Install pip
      apt:
        name: python3-pip
        state: latest
    - name: Install PyMySQL
      shell:
        cmd: python3 -m pip install PyMySQL
    - name: Install mysql server
      apt:
        name: mysql-server
        state: latest
    - name: Stop mysql server
      sysvinit:
        name: mysql
        state: stopped
    - name: Copy credentials file to home directory
      copy:
        src: /etc/mysql/debian.cnf
        remote_src: true
        dest: /home/ansible/.my.cnf
    - name: Create ansible credentials file
      copy:
        src: /etc/mysql/debian.cnf
        remote_src: true
        dest: /home/ansible/.my_ansible.cnf
    - name: Rename user and password items
      shell: |
        sed -i "s/user/login_user/g" /home/ansible/.my_ansible.cnf
        sed -i "s/password/login_password/g" /home/ansible/.my_ansible.cnf
    - name: Allow bind to all hosts
      shell:
        cmd: sed -i -r "s/bind-address\s{1,}= 127.0.0.1/bind-address = 0.0.0.0/" /etc/mysql/mysql.conf.d/mysqld.cnf
    - name: Started and enabled mysql server
      sysvinit:
        name: mysql
        state: started
        enabled: yes
    - name: Create database user
      mysql_user:
        config_file: /home/ansible/.my_ansible.cnf # Login и пароль для входа в БД
        name: db_user_for_app                      # Login нового пользователя
        password: 123                              # Пароль нового пользователя
        host: '%'                                  # Разрешаем заходить с любых хостов
        priv: '*.*:ALL'                          # Даём пользователю все привилегии
        state: present                            # Создать

```

3. Запустите плейбук и дождитесь завершения.

```

ansible@ansible:~/app/ansible$ ansible-playbook install_mysql.yaml

PLAY [Install_mysql] *****

TASK [Gathering Facts] *****
ok: [db]

TASK [Install pip] *****
changed: [db]

TASK [Install PyMySQL] *****
changed: [db]

TASK [Install mysql server] *****
changed: [db]

TASK [Stop mysql server] *****
ok: [db]

TASK [Copy credentials file to home directory] *****
changed: [db]

TASK [Create ansible credentials file] *****
changed: [db]

TASK [Rename user and password items] *****
changed: [db]

TASK [Allow bind to all hosts] *****
changed: [db]

TASK [Started and enabled mysql server] *****
changed: [db]

TASK [Create database user] *****
[WARNING]: Option column_case_sensitive is not provided. The default is now false, so the column's name will be
uppercased. The default will be changed to true in community.mysql 4.0.0.
changed: [db]

PLAY RECAP *****
db                : ok=11  changed=9  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

ansible@ansible:~/app/ansible$

```

Установка софта на рабочие сервера

1. Создайте в каталоге "~/app/ansible" файл "install_worker_soft.yaml";
2. Добавьте в файл следующие строки (с учётом своих ip-адресов):

```

GNU nano 6.2                                install_worker_soft.yaml
- name: Install worker soft
  hosts: workers
  become: yes
  tasks:
    - name: Install python3 pip venv git
      apt:
        name:
          - python3.8
          - python3-pip
          - python3.8-venv
          - git
        state: latest
        update_cache: yes
    - name: Clone Git repository
      git:
        repo: 'https://github.com/VladimirChabanov/url_extender.git'
        dest: /home/ansible/url_extender
        force: yes
    - name: Set data to app config
      shell: |
        sed -i 's/db_host/"192.168.1.200"/' /home/ansible/url_extender/src/config.py
        sed -i 's/user/"db_user_for_app"/' /home/ansible/url_extender/src/config.py
        sed -i 's/passworld/"123"/' /home/ansible/url_extender/src/config.py
        sed -i 's/database/"main"/' /home/ansible/url_extender/src/config.py
        sed -i 's/retries/3/' /home/ansible/url_extender/src/config.py
        sed -i 's/host/"192.168.1.100"/' /home/ansible/url_extender/src/config.py
        sed -i 's/port/80/' /home/ansible/url_extender/src/config.py
    - name: Create virtual env
      shell: python3.8 -m venv venv
    - name: Install requirements into virtual env
      shell: |
        source venv/bin/activate
        pip install --upgrade setuptools wheel pip
        pip install -r /home/ansible/url_extender/requirements.txt
      args:
        executable: /bin/bash
    - name: Run app
      shell: |
        source venv/bin/activate
        if [ -f app_pid ]; then kill -9 `echo app_pid` 2>/dev/null; fi
        nohup python /home/ansible/url_extender/src/run.py </dev/null >/dev/null 2>&1 &
        echo $! > app_pid
      args:
        executable: /bin/bash

```

3. Запустите playbook и дождитесь установки.

```

ansible@ansible:~/app/ansible$ ansible-playbook install_worker_soft.yaml

PLAY [Install worker soft] *****

TASK [Gathering Facts] *****
ok: [worker2]
ok: [worker1]
ok: [worker3]
ok: [worker4]

TASK [Install python3 pip venv git] *****
changed: [worker3]
changed: [worker1]
changed: [worker2]
changed: [worker4]

TASK [Clone Git repository] *****
changed: [worker4]
changed: [worker3]
changed: [worker2]
changed: [worker1]

TASK [Set data to app config] *****
changed: [worker1]
changed: [worker4]
changed: [worker2]
changed: [worker3]

TASK [Create virtual env] *****
changed: [worker3]
changed: [worker4]
changed: [worker1]
changed: [worker2]

TASK [Install requirements into virtual env] *****
changed: [worker4]
changed: [worker3]
changed: [worker2]
changed: [worker1]

TASK [Run app] *****
changed: [worker2]
changed: [worker1]
changed: [worker4]
changed: [worker3]

PLAY RECAP *****
worker1      : ok=7    changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
worker2      : ok=7    changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
worker3      : ok=7    changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
worker4      : ok=7    changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ansible@ansible:~/app/ansible$

```

Установка софта на сервер балансировщика нагрузки

1. Создайте в каталоге "~/app/ansible" файл "install_haproxy.yaml";
2. Добавьте в файл следующие строки (с учётом своих ip-адресов):

```

ansible@ansible:~/app/ansible$ ansible-playbook install_haproxy.yaml

PLAY [Install worker soft] *****

TASK [Gathering Facts] *****
ok: [load_balancer]

TASK [Install haproxy] *****
changed: [load_balancer]

TASK [Stop haproxy] *****
ok: [load_balancer]

TASK [Backup original config] *****
changed: [load_balancer]

TASK [Restore original config from backup] *****
ok: [load_balancer]

TASK [Gen Routing (temporary file)] *****
changed: [load_balancer]

TASK [Join haproxy.cfg и my_routes.cfg] *****
changed: [load_balancer]

TASK [Start and enabled haproxy] *****
changed: [load_balancer]

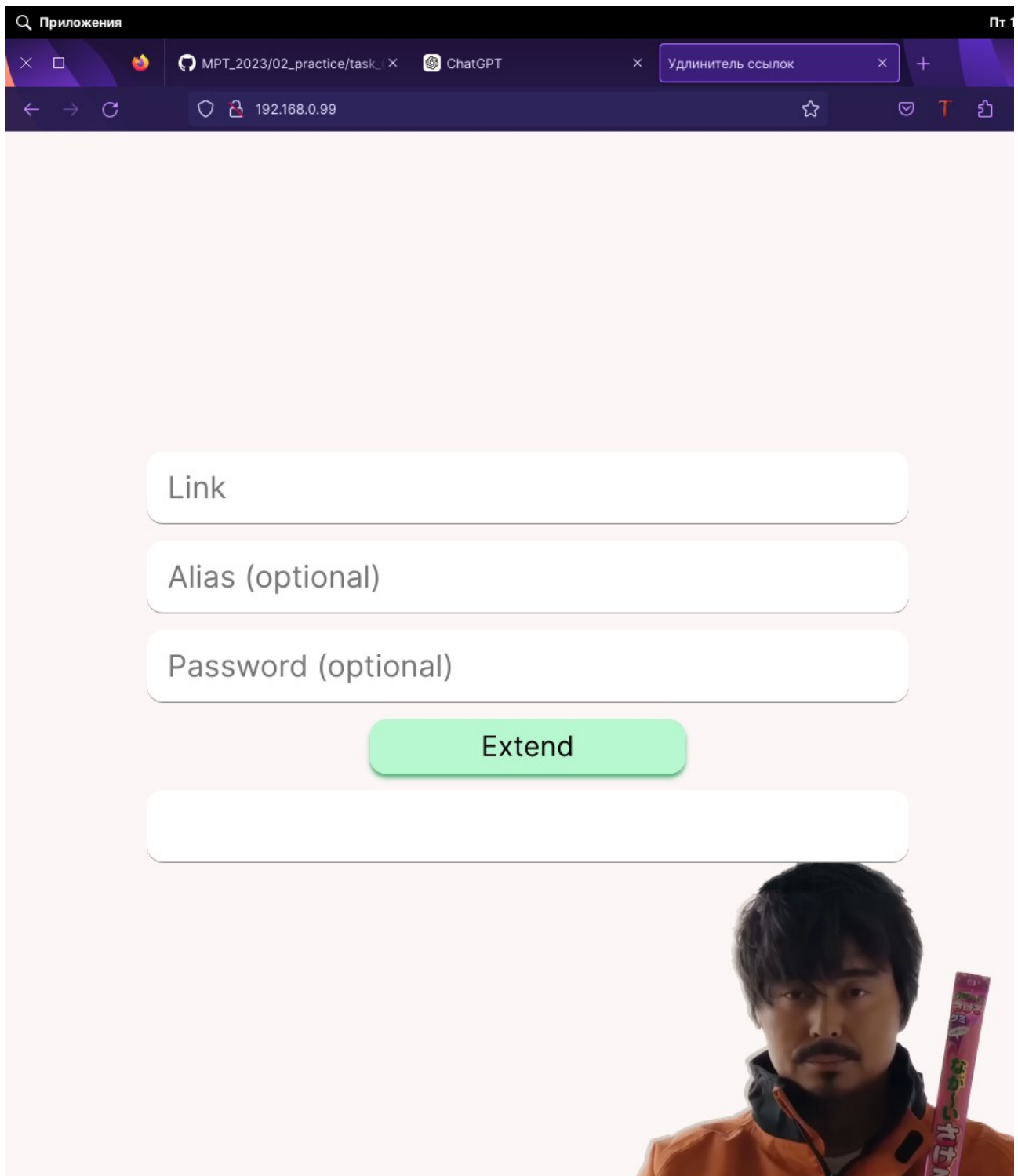
PLAY RECAP *****
load_balancer      : ok=8    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ansible@ansible:~/app/ansible$

```

Теперь нужно убедиться, что приложение развёрнуто на серверах и корректно работает.

1. Откройте браузер на основной операционной системе, или на другом устройстве подключённом к той же сети, и введите ip-адрес сервера "load_balancer" (у меня это 192.168.1.100).



2. Обновите несколько раз страницу и попробуйте воспользоваться функционалом приложения.
3. Зайдите на страницу "/haproxy_stats" (порт 81) и введите учётные данные.
4. Изучите статистику распределения запросов к рабочим серверам столбец **Sessions** -> **Total** в нижней таблице. Количество запросов ко всем серверам должно быть примерно одинаковым, т.к. установлена политика балансировки `roundrobin` (карусель).

pid = 3529 (process #1, nbproc = 1, nbthread = 1)
 uptime = 6d 0h01m47s
 system limits: memmax = unlimited; ulimit-n = 4032
 maxsock = 4032; maxconn = 2000; maxpipes = 0
 current conns = 2; current pipes = 0/0; conn rate = 1/sec
 Running tasks: 1/7; idle = 100 %

active UP, going down
backup UP, going down
active DOWN, going up
backup DOWN, going up
active or backup DOWN
not checked
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
 Note: "NO LB"/"DRAIN" = UP with load-balancing disabled.

Scope:
[Hide DOWN servers](#)
[Refresh now](#)
[CSV export](#)
[Primary site](#)
[V060606.v1.0](#)
[Online manual](#)

stats		Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Server													
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtime	Thrtle		
Frontend		0	0	0	1	2	-	1	2	2 000	7			5 010	137 645	0	0	1						OPEN									
Backend		0	0	0	0	1		0	1	200	5	0	0s	5 010	137 645	0	0		5	0	0	0	1m47s UP		0	0	0		0				

my-web

		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtime	Thrtle	
Frontend		0	0	0	1	2	-	1	2	2 000	26			80 909	18 586 103	0	0	0					OPEN									

my-web

		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtime	Thrtle
<input type="checkbox"/>	myweb1	0	0	-	0	2		0	1	-	18	18	2s	9 504	3 710 057	0	0	0	0	0	0	0	no check		1	Y	-				-
<input type="checkbox"/>	myweb2	0	0	-	0	2		0	2	-	18	18	2s	9 091	3 068 448	0	0	0	0	0	0	0	no check		1	Y	-				-
<input type="checkbox"/>	myweb3	0	0	-	0	2		0	1	-	18	18	2s	8 637	1 497 699	0	0	0	0	0	0	0	no check		1	Y	-				-
<input type="checkbox"/>	myweb5	0	0	-	0	2		0	1	-	17	17	3s	8 409	2 164 788	0	0	0	0	0	0	0	no check		1	Y	-				-
<input type="checkbox"/>	myweb6	0	0	-	0	2		0	1	-	17	17	3s	8 668	1 427 232	0	0	0	0	0	0	0	no check		1	Y	-				-
<input type="checkbox"/>	myweb7	0	0	-	0	2		0	1	-	17	17	2s	8 326	1 499 938	0	0	0	0	0	0	0	no check		1	Y	-				-
<input type="checkbox"/>	myweb8	0	0	-	0	2		0	1	-	17	17	2s	8 214	5 187 941	0	0	0	0	0	0	0	no check		1	Y	-				-
	Backend	0	0	0	0	14		0	5	200	122	122	2s	80 909	18 586 103	0	0	0	0	0	0	0	1m47s UP		7	7	0		0	0s	

Choose the action to perform on the checked servers:

Теперь увеличим количество рабочих серверов.

1. Перейдите на виртуальную машину с docker compose и остановите запущенные контейнеры;

```
internet@internet:~/compose$ docker compose down
[+] Running 7/6
✓ Container compose-worker1-1      Removed
✓ Container compose-worker2-1      Removed
✓ Container compose-worker3-1      Removed
✓ Container compose-worker4-1      Removed
✓ Container compose-load_balancer-1 Removed
✓ Container compose-db-1           Removed
✓ Network compose_outside          Removed
internet@internet:~/compose$
```

2. Добавьте в файл compose.yaml дополнительные сервера, таки образом, чтобы рабочих серверов было 8;

```
services:
  load_balancer: # Балансировщик нагрузки
    image: rastasheep/ubuntu-sshd
    dns:
      - 77.88.8.8
      - 8.8.8.8
    networks:
      outside:
        ipv4_address: 192.168.0.99
  db: # База данных
    image: rastasheep/ubuntu-sshd
    dns:
      - 77.88.8.8
      - 8.8.8.8
    networks:
      outside:
        ipv4_address: 192.168.0.98
  worker1: # Сервер с приложением
    image: rastasheep/ubuntu-sshd
    dns:
      - 77.88.8.8
      - 8.8.8.8
    networks:
      outside:
        ipv4_address: 192.168.0.97
  worker2: # Сервер с приложением
    image: rastasheep/ubuntu-sshd
    dns:
      - 77.88.8.8
      - 8.8.8.8
    networks:
      outside:
        ipv4_address: 192.168.0.96
  worker3: # Сервер с приложением
    image: rastasheep/ubuntu-sshd
    dns:
      - 77.88.8.8
      - 8.8.8.8
    networks:
      outside:
        ipv4_address: 192.168.0.95
  worker4: # Сервер с приложением
    image: rastasheep/ubuntu-sshd
    dns:
      - 77.88.8.8
      - 8.8.8.8
    networks:
```

3. Запустите их при помощи `docker compose up -d`. Все машины будут чистыми с настройками по умолчанию;

```
internet@internet:~/compose$ docker compose up -d
[+] Running 11/11
 ✓ Network compose_outside          Created
 ✓ Container compose-worker1-1      Started
 ✓ Container compose-worker2-1      Started
 ✓ Container compose-worker7-1      Started
 ✓ Container compose-load_balancer-1 Started
 ✓ Container compose-db-1           Started
 ✓ Container compose-worker3-1      Started
 ✓ Container compose-worker5-1      Started
 ✓ Container compose-worker8-1      Started
 ✓ Container compose-worker6-1      Started
 ✓ Container compose-worker4-1      Started
internet@internet:~/compose$
```

4. Добавьте новые сарваера в файл `hosts` (учтите, что теперь там снова пользователь `root`), запустите начальное конфигурирование и установку всего необходимого софта;
5. В конфиге балансировщика, в раздел `"backend"` добавьте новые рабочие сервера;
6. Протестируйте работоспособность приложения, можно просто обновить вкладку раз 20-30;
7. Зайдите на страницу `"/haproxy_stats"` и проверьте, что все сервера работают и загружены равномерно.

Ответы на вопросы.

1. Команде ``ansible-playbook`` можно указать файл с серверами (`hosts`) несколькими способами:

а) Указать имя файла с хостами через параметр ``-i`` (`inventory`):

...

```
ansible-playbook -i hosts_file.txt playbook.yml
```

...

Здесь ``hosts_file.txt`` - файл с хостами, а ``playbook.yml`` - файл с плейбуком.

б) Указать имя файла с хостами в конфигурационном файле ``ansible.cfg``:

...

```
[defaults]
```

```
inventory = hosts_file.txt
```

...

В этом случае можно выполнять команду без указания параметра ``-i``:

...

```
ansible-playbook playbook.yml
```

...

2. При условии, что доступ по `ssh` уже есть, на Linux-серверах, которыми планируется управлять при помощи Ansible, нужно установить программное

обеспечение, которое поддерживает Ansible. Это, в основном, Python. Python должен быть установлен на серверах, а также на компьютере, с которого выполняются команды Ansible.

3. В логах выполнения playbook-а показатель "changed" в разделе "PLAY RECAP" означает, что состояние хоста было изменено. Это может быть изменение файлов, установка пакетов и другие операции. Если значение "changed" равно 0, значит состояние хоста не было изменено.

4. Ad hoc команда ansible для создания пользователя "thor", который состоит только в группах "avengers" и "asgard" с паролем "Strongest Avenger" на хостах из группы "stark_industries":

```
...  
ansible stark_industries -m user -a "name=thor groups='avengers,asgard'  
password=Strongest Avenger"  
...
```

Здесь `stark_industries` - группа хостов, `user` - модуль для работы с пользователями, `name`, `groups` и `password` - параметры модуля.

5. Playbook для удаления половины файлов из каталога "/lifeforms" на серверах из группы "universe":

```
...  
---  
- hosts: universe  
  tasks:  
    - name: Delete half of the files  
      file:  
        path: "/lifeforms/{{ item }}"  
        state: absent  
        when: item.endswith('1') or item.endswith('2')  
        loop: "{{ query('fileglob', '/lifeforms/*') }}"  
        when: query('fileglob', '/lifeforms/*') | length % 2 == 0  
...
```

Здесь `hosts: universe` указывает, что плейбук будет выполнен на серверах из группы "universe". `file` - модуль для работы с файлами, `path: "/lifeforms/{{ item }}"` - путь к файлу, `state: absent` - удаляем файл. `when: item.endswith('1') or item.endswith('2')` - условие для удаления файлов, заканчивающихся на 1 или 2. `loop: "{{ query('fileglob', '/lifeforms/*') }}"` - цикл по всем файлам в каталоге "/lifeforms". `when: query('fileglob', '/lifeforms/*') | length % 2 == 0` - условие для проверки количества файлов в каталоге, оно должно быть чётным.

Вывод:

Я ознакомился на практике с инструментом для удаленного управления конфигурациями Ansible.