



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных
технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 5
по дисциплине
«Технология разработки программных приложений»

Тема: «Ansible»

Выполнил студент группы ИНБО-12-23

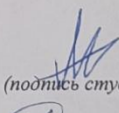
Албахтин И.В.

Принял

Петренко А.А.

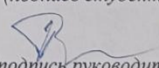
Практическая работа выполнена

«14» мая 2025г.


(подпись студента)

«Зачтено»

«14» мая 2025 г.


(подпись руководителя)

Москва 2025

Выполнение практической работы

Цель работы: получить навыки настройки вычислительной инфраструктуры при помощи системы конфигурационного управления Ansible.

1. Подготовка инфраструктуры

Прежде всего, необходимо создать 2 виртуальные машины Ubuntu или Debian. Этими машинами необходимо будет управлять в ходе практики. Имя пользователя должно быть указано как user. Это важно для дальнейшей работы.

После заполнения перезапустите машину командой `reboot`. Ещё одним важным шагом является установка ssh-сервера, который позволит удалённо подключаться к машине. Установить его можно при помощи команды:

```
apt install ssh
```

Также необходимо создать управляющую машину, на которой непосредственно будет установлен Ansible. Это может быть как виртуальная машина, тогда необходимо будет проделать те же самые манипуляции, что и для управляемых машин, так и хостовая машина. Для пользователей Windows рекомендуется воспользоваться WSL2, который позволяет поднять Linux прямо на хостовой машине. Работа проводится на Windows, WSL 2, Ubuntu.

Первый эмулированный пользователь находится на 22 порту, fed1, рис 1.

```
-> ~ ssh -p 22 mattew@127.0.0.1
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 5.15.167.4-microsoft-standard-WSL2 x86_64)

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

System information as of Wed Apr 23 21:10:35 MSK 2025

System load:  0.0               Processes:    130
Usage of /:   5.6% of 250.92GB  Users logged in: 0
Memory usage: 12%              IPv4 address for eth0: 172.21.170.65
Swap usage:  0%

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
  just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge
Last login: Tue Apr 22 16:46:34 2025 from 127.0.0.1
```

Рисунок 1 – Первая виртуальная машина, подключение 22 порт

Второй эмулированный пользователь находится на 2222, fed2, рис 2.

```
-> ~ ssh -p 2222 mattew@127.0.0.1
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 5.15.167.4-microsoft-standard-WSL2 x86_64)

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

System information as of Wed Apr 23 21:20:10 MSK 2025

System load:  0.19           Processes:            176
Usage of /:   5.7% of 250.92GB Users logged in:          1
Memory usage: 62%           IPv4 address for eth0: 172.21.170.65
Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge
Last login: Wed Apr 23 21:10:50 2025 from 127.0.0.1
```

Рисунок 2 – Вторая виртуальная машина, подключение 2222 порт

2. Установка Ansible

Для начала необходимо установить сам Ansible на управляющую машину. Сделать это можно при помощи следующих команд: `sudo apt install ansible` - будет установлена версия Ansible, содержащаяся в текущем выпуске дистрибутива `sudo pip3 install ansible` - установка при помощи пакетного менеджера Python 3 (поскольку Ansible написан на Python).

Для установки через `pip` может потребоваться его установить. Сделать это можно командой `sudo apt install python3-pip`. Наличие Ansible можно проверить командой `ansible --version`. После чего попробуем подключиться к управляемым машинам при помощи `ssh`. `ssh root@ip_address` где `ip_address` - то адрес управляемого узла в сети, рис 3.

```
-> ~ ansible --version
ansible [core 2.16.3]
  config file = None
  configured module search path = ['/home/mattew/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/mattew/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Feb  4 2025, 14:48:35) [GCC 13.3.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
```

Рисунок 3 – Отладка работы ansible на хостовой машине

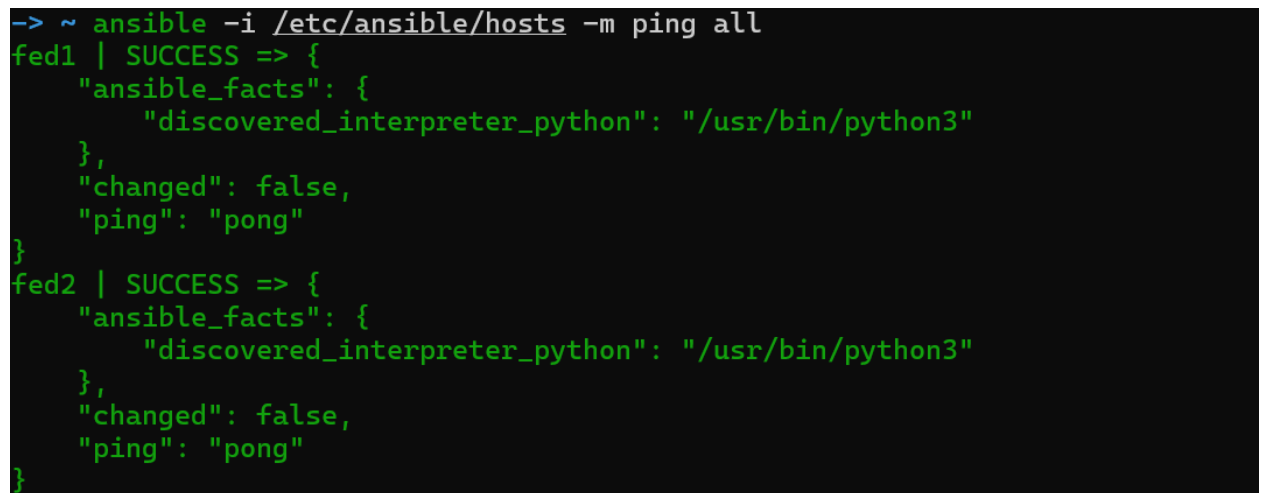
3. Настройка Ansible

Все действия также необходимо выполнять из-под пользователя `root`. Настроим `inventory`-файл. Создадим директорию `ansible` в домашнем каталоге пользователя для хранения репозитория инфраструктуры, будем считать эту

директорию рабочей, и в ней создадим файл `hosts`. Заполните файл по следующему образцу. Inventory-файл содержит в себе всю необходимую информацию о хостах, которые должны управляться системой Ansible. Выполним самую первую команду для проверки работы Ansible:

```
ansible -i ./hosts -m ping all
```

Данная команда выполнит команду `ping` для всех хостов в inventory и выдаст результат выполнения, рис 4.



```
--> ~ ansible -i /etc/ansible/hosts -m ping all
fed1 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
fed2 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
```

Рисунок 4 – Проверка подключения к виртуальным машинам

4. Использование Ansible для конфигурации хостов

Установка Ansible

Ansible — это инструмент автоматизации без агентов, который вы устанавливаете на одном хосте (называемом узлом управления). С управляющего узла Ansible может удаленно управлять целым парком машин и других устройств (называемых управляемыми узлами) с помощью SSH, удаленного управления Powershell и множества других транспортных средств, и все это с помощью простого интерфейса командной строки, не требующего баз данных или демонов.

Факты

Факты — это параметры, которыми можно управлять в реализуемых сценариях.

Соберем возможные факты с управляемого хоста `fed1`, рис. 5:

```
ansible fed1 -i ./hosts -m command -m setup
```

```

-> ~ ansible fed1 -i /etc/ansible/hosts -m setup
fed1 | SUCCESS => {
  "ansible_facts": {
    "ansible_all_ipv4_addresses": [
      "172.17.0.1",
      "10.255.255.254",
      "172.21.170.65"
    ],
    "ansible_all_ipv6_addresses": [
      "fe80::215:5dff:fe24:3396"
    ],
    "ansible_apparmor": {
      "status": "disabled"
    },
    "ansible_architecture": "x86_64",
    "ansible_bios_date": "NA",
    "ansible_bios_vendor": "NA",
    "ansible_bios_version": "NA",
    "ansible_board_asset_tag": "NA",
    "ansible_board_name": "NA",
    "ansible_board_serial": "NA",
    "ansible_board_vendor": "NA",
    "ansible_board_version": "NA",
    "ansible_chassis_asset_tag": "NA",
    "ansible_chassis_serial": "NA",
    "ansible_chassis_vendor": "NA",
    "ansible_chassis_version": "NA",
    "ansible_cmdline": {
      "WSL_ENABLE_CRASH_DUMP": "1",
      "WSL_ROOT_INIT": "1",
      "console": "hvc0",

```

Рисунок 5 – Факты машины fed1

Playbook

Это конфигурационный сценарий, написанный на языке YAML, который впоследствии будет выполняться на управляемых хостах.

Напишем playbook, который установит веб-сервер Nginx на управляемые хосты.

Сам файл с playbook'ОМ состоит из сценариев. Каждый сценарий начинается с ключевого слова `name`.

`tasks` — это список задач, которые необходимо выполнить на управляемой машине в ходе сценария. Каждый элемент списка задач начинается с символа дефис (-). Этот символ в языке YAML обозначает начало ассоциативного массива.

При помощи ключевого слова `name` задается имя задачи. При помощи ключевого слова `when` задаются условия для запуска задачи, к примеру в указанном примере задаётся условие, что задача по установке `nginx` должна выполняться только на хосте под управлением Debian.

`handlers` — это действия, которые будут выполняться после завершения

задачи. Они запускаются на выполнения только 1 раз и после завершения всего сценария. Ключевое слово `notify` позволяет запускать `handler`'ы. В указанном примере после установки `nginx` будет запущен `handler`, активирующий автозапуск сервиса `nginx`.

После написания `playbook`'а его можно выполнить при помощи следующей команды `ansible-playbook -i hosts`

В результате успешного запуска вы должны увидеть следующий вывод в терминале, рис 6.

```
-> trpp-ansible ansible-playbook ansible1.yml --ask-become-pass
BECOME password:

PLAY [Install Nginx to Webservers] *****

TASK [Gathering Facts] *****
ok: [fed1]
ok: [fed2]

TASK [Install Nginx] *****
ok: [fed2]
ok: [fed1]

PLAY RECAP *****
fed1                : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
fed2                : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Рисунок 6 – Результат выполнения `playbook`'а

После этого можно перейти на любой из двух узлов (при помощи `ssh` или же непосредственно открыв окно виртуальной машины) и убедиться в том, что `nginx` действительно установлен при помощи команды `systemctl status nginx`.

Теперь остановим и удалим с данного узла `nginx` при помощи следующих команд непосредственно через терминал.

`systemctl stop nginx`

`apt remove nginx`

Теперь повторно необходимо запустить `playbook`, рис 7

```
-> trpp-ansible ansible-playbook ansible1.yml --ask-become-pass
BECOME password:

PLAY [Install Nginx to Webservers] *****

TASK [Gathering Facts] *****
ok: [fed1]
ok: [fed2]

TASK [Install Nginx] *****
ok: [fed2]
ok: [fed1]

PLAY RECAP *****
fed1                : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
fed2                : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Рисунок 7 – Результат повторного выполнения `playbook`'а

Более сложный playbook

Начнем с написания более сложного playbook'а и рассмотрим отдельные его составляющие:

```
---
- name: Install and config Nginx
  hosts: all
  become: yes

  vars:
    html_dir: /usr/share/nginx/html
    greeting: "Hello Everybody!"

  tasks:
    - name: Install Nginx
      apt:
        name: nginx
        state: present
        update_cache: yes
      notify:
        - Nginx Systemd

    - name: Delete default HTML files
      shell: /bin/rm -rf /usr/share/nginx/html/*.html

    - name: Replace config file
      vars:
        nginx_user: user
        worker_processes: 2
        worker_connections: 256
      template:
        src: templates/nginx.conf.j2
        dest: /etc/nginx/nginx.conf
        mode: 0644
      register: result
      failed_when: result.failed == true
      notify: Reload Nginx

    - name: Copy index file
      copy:
        src: files/index.html
        dest: {{ html_dir }}
        mode: 0644
      notify: Reload Nginx
```



```
- name: Generate dynamic HTML from template
  template:
    src: templates/hello.html.j2
    dest: {{ html_dir }}/hello.html
    owner: root
    mode: 0644
    notify: Reload Nginx

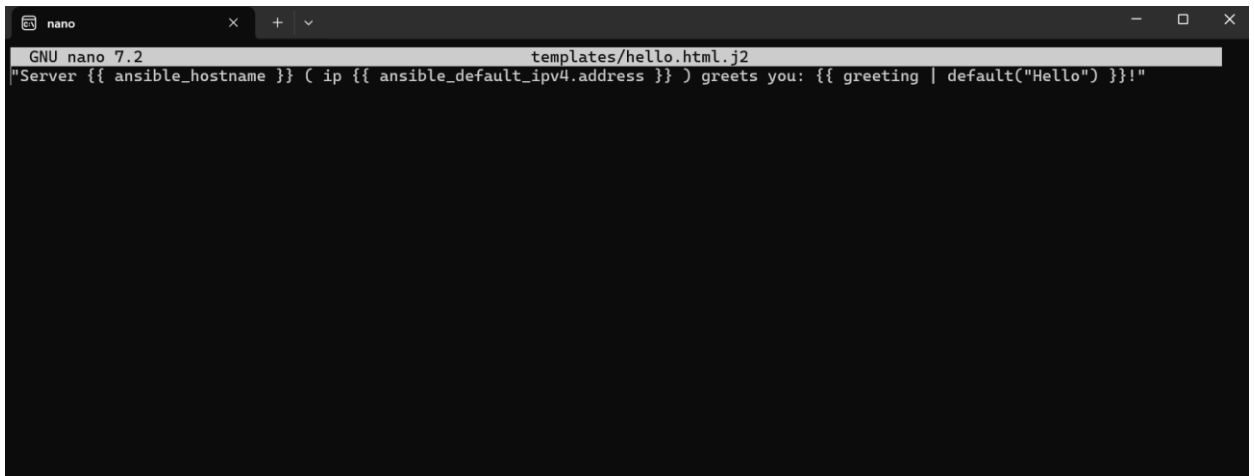
handlers:
- name: Nginx Systemd
  systemd:
    name: nginx
    enabled: yes
    state: started

- name: Reload Nginx
  systemd:
    name: nginx
    state: reloaded
```

В разделе `vars` описываются переменные, которые затем можно использовать при запуске задач. Для использования переменных в задачах используется синтаксис шаблонизатора - `{{ имя_переменной }}`.

Ещё одним инструментом является использование шаблонных файлов. Для этого используется шаблонизатор Jinja2. Шаблонные файлы добавляются в директорию `templates` с дополнительным расширением `j2`, то есть итоговое имя файла должно выглядеть, например, следующим образом `index.html.j2`. При реализации вышеописанного `playbook`'а используется 2 шаблонных файла: `nginx.conf.j2`, `hello.html.j2`.

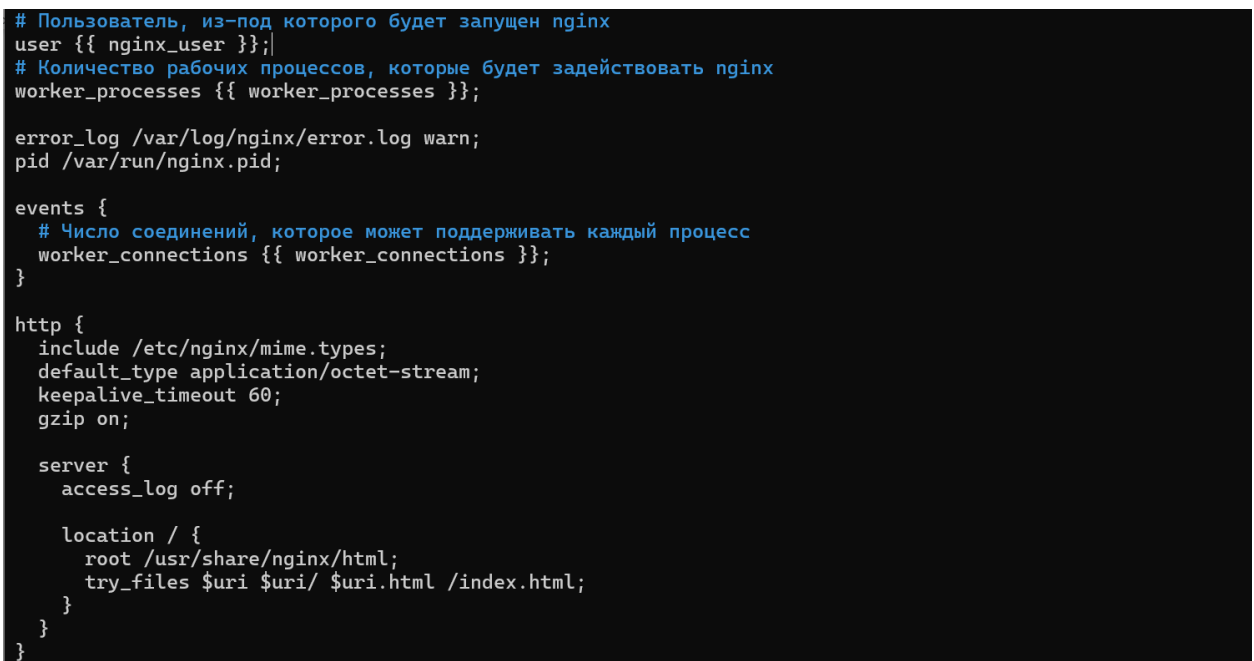
Содержимое файла `hello.html.j2`, рис 8.



```
GNU nano 7.2 templates/hello.html.j2
"Server {{ ansible_hostname }} ( ip {{ ansible_default_ipv4.address }} ) greets you: {{ greeting | default("Hello") }}!"
```

Рисунок 8 – Файл hello.html.j2

Содержимое файла nginx.conf.j2, рис 9.



```
# Пользователь, из-под которого будет запущен nginx
user {{ nginx_user }};|
# Количество рабочих процессов, которые будет задействовать nginx
worker_processes {{ worker_processes }};

error_log /var/log/nginx/error.log warn;
pid /var/run/nginx.pid;

events {
    # Число соединений, которое может поддерживать каждый процесс
    worker_connections {{ worker_connections }};
}

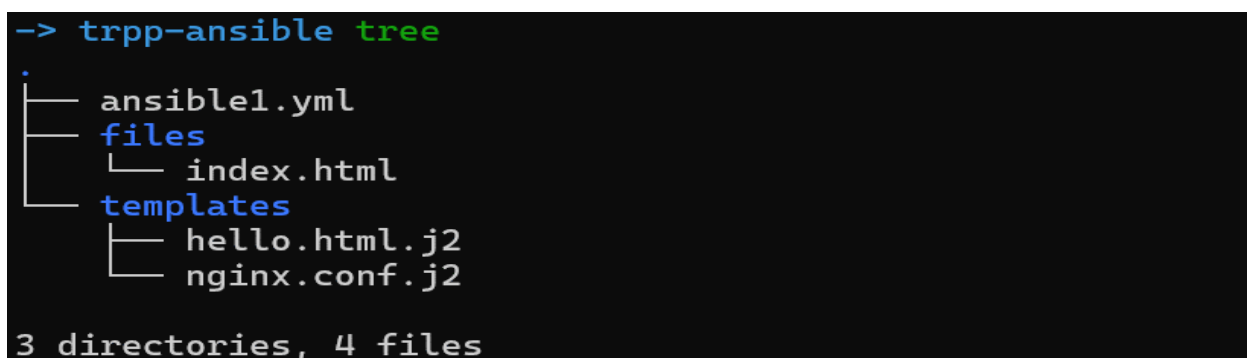
http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;
    keepalive_timeout 60;
    gzip on;

    server {
        access_log off;

        location / {
            root /usr/share/nginx/html;
            try_files $uri $uri/ $uri.html /index.html;
        }
    }
}
```

Рисунок 9 – Файл nginx.conf.j2

В результате содержимое директории должно иметь следующий вид, рис 10.



```
-> trpp-ansible tree
.
├── ansible1.yml
├── files
│   └── index.html
└── templates
    ├── hello.html.j2
    └── nginx.conf.j2

3 directories, 4 files
```

Рисунок 10 – Содержимое директории проекта

Теперь выполним команду для запуска playbook'a с пробным прогоном, который позволит проверить корректность написанного playbook'a без внесения изменений на целевые узлы, рис 11

ansible-playbook -i hosts --check

```
-> trpp-ansible ansible-playbook ansible1.yml --ask-become-pass
BECOME password:

PLAY [Install and config Nginx] *****

TASK [Gathering Facts] *****
ok: [fed1]
ok: [fed2]

TASK [Install Nginx] *****
ok: [fed2]
ok: [fed1]

TASK [Delete default HTML files] *****
changed: [fed1]
changed: [fed2]

TASK [Replace config file] *****
ok: [fed1]
ok: [fed2]

TASK [Copy index file] *****
changed: [fed2]
ok: [fed1]

TASK [Generate dynamic HTML from template] *****
changed: [fed1]
ok: [fed2]

RUNNING HANDLER [Reload Nginx] *****
changed: [fed2]
changed: [fed1]

PLAY RECAP *****
fed1                : ok=7    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
fed2                : ok=7    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
-> trpp-ansible curl http://localhost
Web server is working!
```

Рисунок 11 – Результат выполнения плейбука

Теперь внесём изменения в конфигурацию первого сервера, убрав ключ --check из предыдущей команды, а затем запросим базовую страницу при помощи всё той же утилиты curl.

curl -L http://127.0.0.1

Также можно запросить созданную при помощи шаблона страницу hello.

curl -L http://127.0.0.1/hello

Как можно увидеть, рис 13, изменения успешно применились. Можете самостоятельно поменять переменную greetings, затем загрузить повторно playbook на машину и понаблюдать за изменениями.

После этого применим playbook для всех серверов в группе, рис 12.

```

BECOME password:

PLAY [Install and config Nginx] *****

TASK [Gathering Facts] *****
ok: [fed1]
ok: [fed2]

TASK [Install Nginx] *****
ok: [fed2]
ok: [fed1]

TASK [Delete default HTML files] *****
changed: [fed1]
changed: [fed2]

TASK [Replace config file] *****
ok: [fed1]
ok: [fed2]

TASK [Copy index file] *****
changed: [fed2]
ok: [fed1]

TASK [Generate dynamic HTML from template] *****
changed: [fed1]
ok: [fed2]

RUNNING HANDLER [Reload Nginx] *****
changed: [fed2]
changed: [fed1]

PLAY RECAP *****
fed1                : ok=7    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
fed2                : ok=7    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

```

Рисунок 12 – Проверка работы плейбука

Как можно заметить, при небольшом расширении задачи, решаемой при помощи Ansible, файл playbook'a начинает разрастаться, что вызывает неудобство при его чтении.

```

-> trpp-ansible curl http://127.0.0.1
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
-> trpp-ansible curl http://127.0.0.1/hello
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.24.0 (Ubuntu)</center>
</body>
</html>

```

Рисунок 13 – Просмотр локального сайта конфигурации Nginx

Роли Ansible

Данный механизм позволяет систематизировать конфигурации путём выделения каждого механизма (задач, переменных, обработчиков и т. д.) в отдельные части.

Ansible имеет похожий на GitHub сервис, называемый Ansible Galaxy. Там находится множество ролей для Ansible, которыми можно воспользоваться. Для использования данного сервиса в Ansible встроена команда `ansible-galaxy`. Загрузим в систему роль для установки MySQL при помощи команды

```
ansible-galaxy install geerlingguy.mysql
```

Найти файлы установленной роли можно по следующему пути `~/.ansible/roles/geerlingguy.mysql/`. Рассмотрим более подробно содержимое данной роли. При помощи команды `tree` можно вывести дерево каталога для большей наглядности, рис 14.

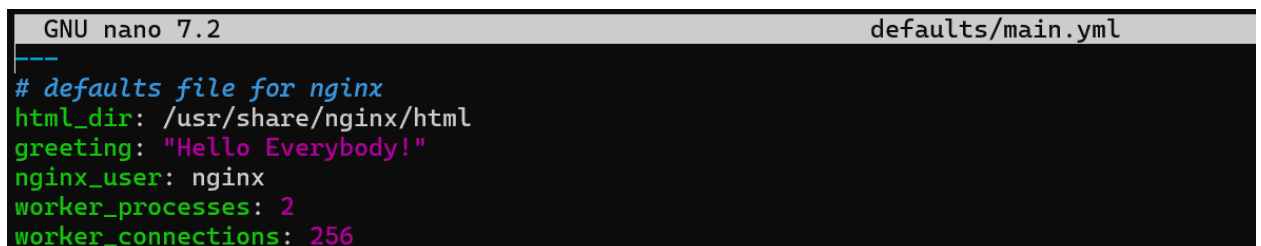
```
-> trpp-ansible tree ~/.ansible/roles/geerlingguy.mysql/
.
├── ansible1.yml
├── files
│   └── index.html
├── templates
│   ├── hello.html.j2
│   └── nginx.conf.j2
└── /home/mattew/.ansible/roles/geerlingguy.mysql/
    ├── LICENSE
    ├── README.md
    ├── defaults
    │   └── main.yml
    ├── handlers
    │   └── main.yml
    ├── meta
    │   └── main.yml
    ├── molecule
    │   └── default
    │       ├── converge.yml
    │       └── molecule.yml
    ├── tasks
    │   ├── configure.yml
    │   ├── databases.yml
    │   ├── main.yml
    │   ├── replication.yml
    │   ├── secure-installation.yml
    │   ├── setup-Archlinux.yml
    │   ├── setup-Debian.yml
    │   ├── setup-RedHat.yml
    │   ├── users.yml
    │   └── variables.yml
    ├── templates
    │   ├── my.cnf.j2
    │   ├── root-my.cnf.j2
    │   └── user-my.cnf.j2
    └── vars
        ├── Archlinux.yml
        ├── Debian-10.yml
        ├── Debian-11.yml
        ├── Debian-12.yml
        ├── Debian.yml
        ├── RedHat-7.yml
        ├── RedHat-8.yml
        ├── RedHat-9.yml
        └── Ubuntu.yml

12 directories, 33 files
```

Рисунок 14 – Содержимое директории с ролью `mysql`

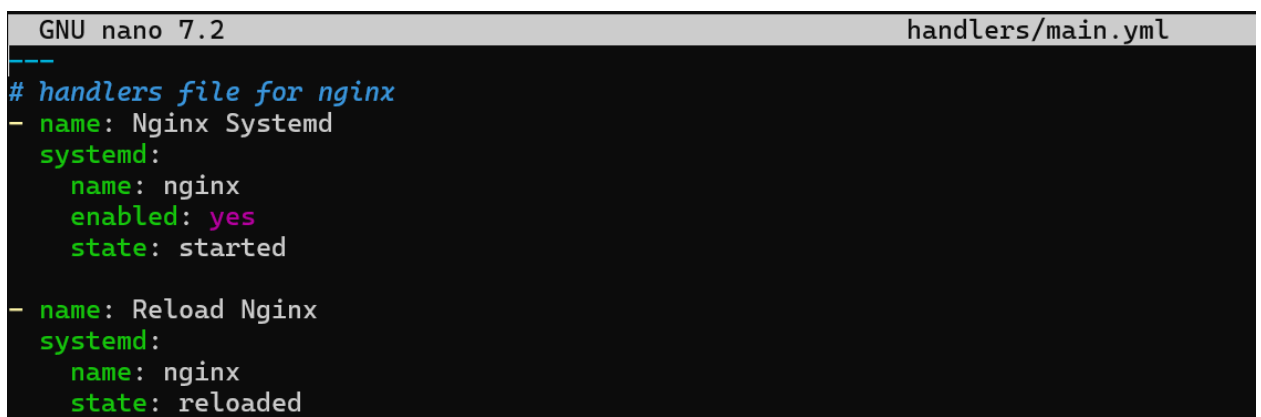
- Директория defaults содержит значения переменных по умолчанию.
- Директория handlers содержит описание обработчиков.
- Директория meta содержит информацию о роли, то есть создателе роли, её описание, используемая лицензия, зависимости и т. д. Данный файл используется в дальнейшем для Ansible Galaxy, который будет описан далее.
- Директория molecule содержит сценарии тестирования для роли Ansible.
- Директория tasks содержит непосредственно сценарии для конфигураций.
- Директория templates, как можно догадаться, содержит шаблонные параметризованные файлы.
- Директория vars содержит файлы описания различных переменных.

Это значит, что в директории tasks файл main.yml должен быть заполнен данными из секции tasks(рис 17). В директории defaults - из секции vars(рис 15). В директории handlers - из секции handlers(рис 16). Также файлы из директорий files и templates должны быть перемещены в директории files и templates в папке роли, рис 18.



```
GNU nano 7.2 defaults/main.yml
---
# defaults file for nginx
html_dir: /usr/share/nginx/html
greeting: "Hello Everybody!"
nginx_user: nginx
worker_processes: 2
worker_connections: 256
```

Рисунок 15 – Файл с переменными роли



```
GNU nano 7.2 handlers/main.yml
---
# handlers file for nginx
- name: Nginx Systemd
  systemd:
    name: nginx
    enabled: yes
    state: started
- name: Reload Nginx
  systemd:
    name: nginx
    state: reloaded
```

Рисунок 16 – Файл с handlers роли

```
GNU nano 7.2 tasks/main.yml
# tasks file for nginx
- name: Install Nginx
  apt:
    name: nginx
    state: present
    update_cache: yes
    notify: Nginx Systemd

- name: Delete default HTML files
  shell: /bin/rm -rf {{ html_dir }}/*.html

- name: Replace config file
  template:
    src: nginx.conf.j2
    dest: /etc/nginx/nginx.conf
    mode: 0644
  register: result
  failed_when: result.failed == true
  notify: Reload Nginx

- name: Copy index file
  copy:
    src: index.html
    dest: "{{ html_dir }}"
    mode: 0644
  notify: Reload Nginx

- name: Generate dynamic HTML from template
  template:
    src: hello.html.j2
    dest: "{{ html_dir }}/hello.html"
    owner: root
    mode: 0644
  notify: Reload Nginx
```

Рисунок 17 – Файл с tasks роли

```
-> nginx ls
README.md defaults files handlers meta tasks templates tests vars
-> nginx tree
.
├── README.md
├── defaults
│   └── main.yml
├── files
│   └── index.html
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── tasks
│   └── main.yml
├── templates
│   ├── hello.html.j2
│   └── nginx.conf.j2
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml

9 directories, 11 files
```

Рисунок 18 – Файловая система проекта

Результат выполнения роли отображен на рис 19.

```

-> trpp-ansible ansible-playbook ansible1.yml --ask-become-pass
BECOME password:

PLAY [Install and config Nginx] *****

TASK [Gathering Facts] *****
ok: [fed1]
ok: [fed2]

TASK [Install Nginx] *****
ok: [fed2]
ok: [fed1]

TASK [Delete default HTML files] *****
changed: [fed1]
changed: [fed2]

TASK [Replace config file] *****
ok: [fed1]
ok: [fed2]

TASK [Copy index file] *****
changed: [fed2]
ok: [fed1]

TASK [Generate dynamic HTML from template] *****
changed: [fed1]
ok: [fed2]

RUNNING HANDLER [Reload Nginx] *****
changed: [fed2]
changed: [fed1]

PLAY RECAP *****
fed1      : ok=7    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
fed2      : ok=7    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
-> trpp-ansible curl http://localhost
Web server is working!

```

Рисунок 19 – Результат выполнения плейбука и роли

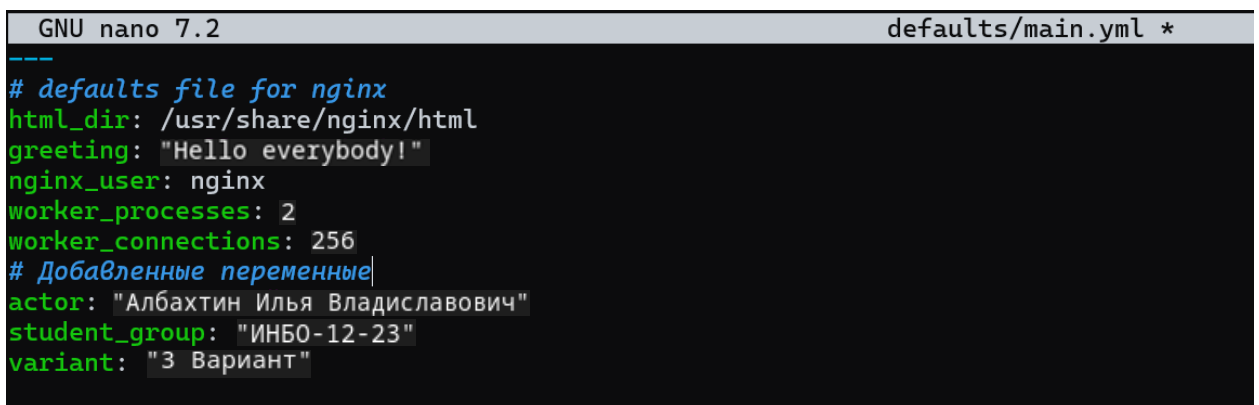
5. Индивидуальное задание

Написать роль для запуска сервера nginx, написать playbook для применения роли, провести тестовый запуск playbook'а, в случае успешного прохождения теста, применить playbook к серверам.

Необходимо добавить переменную, содержащую ФИО, номер группы и номер варианта (Албахтин Илья Владиславович, ИНБО-12-23, 3 Вариант). Данная переменная должна выводиться в шаблонный файл nginx.

Установка пакета выполняется при помощи модуля APT, используемого для установки nginx в базовой роли.

Добавьте в playbook task по установке пакета согласно варианту, 3. wget
Измененный файл main.yml в директории defaults отображен на рис. 20.



```
GNU nano 7.2                                defaults/main.yml *
---
# defaults file for nginx
html_dir: /usr/share/nginx/html
greeting: "Hello everybody!"
nginx_user: nginx
worker_processes: 2
worker_connections: 256
# Добавленные переменные
actor: "Албахтин Илья Владиславович"
student_group: "ИНБО-12-23"
variant: "3 Вариант"
```

Рисунок 20 – Новый файл defaults/main.yml

Измененный файл main.yml в директории tasks, рис 21.

```

GNU nano 7.2                                tasks/main.yml *
---
# tasks file for nginx
- name: Install Nginx
  apt:
    name: nginx
    state: present
    update_cache: yes
  notify: Nginx Systemd

# Добавленный новый компонент установки пакета
- name: Install a unique package
  apt:
    name: wget
    state: present
    update_cache: yes

- name: Delete default HTML files
  shell: /bin/rm -rf {{ html_dir }}/*.html

- name: Replace config file
  template:
    src: nginx.conf.j2
    dest: /etc/nginx/nginx.conf
    mode: 0644
  register: result
  failed_when: result.failed == true
  notify: Reload Nginx

- name: Copy index file
  copy:
    src: index.html
    dest: "{{ html_dir }}"
    mode: 0644
  notify: Reload Nginx

- name: Generate dynamic HTML from template
  template:
    src: hello.html.j2

```

Рисунок 21 – Установка пакета wget

Для удобства и автоматизации использовалась утилита molecule, проводящая тесты роли и плейбука соответственно, рис 22.

```

(molecule_ansible) -> newrole (main)molecule test
WARNING Driver podman does not provide a schema.
INFO default scenario test matrix: dependency, cleanup, destroy, syntax, create, prepare, converge, idempotence, side_effect, verify, cleanup, destroy
INFO Performing prerun with role_name_check=0...
INFO Running default > dependency
WARNING Skipping, missing the requirements file.
WARNING Skipping, missing the requirements file.
INFO Running default > cleanup
WARNING Skipping, cleanup playbook not configured.
INFO Running default > destroy
INFO Sanity checks: 'podman'

PLAY [Destroy] *****

TASK [Set async_dir for HOME env] *****
ok: [localhost]

TASK [Destroy molecule instance(s)] *****
changed: [localhost] => (item={'image': 'quay.io/centos/centos:stream8', 'name': 'instance', 'pre_build_image': True})

TASK [Wait for instance(s) deletion to complete] *****
FAILED - RETRYING: [localhost]: Wait for instance(s) deletion to complete (300 retries left).
changed: [localhost] => (item={'failed': 0, 'started': 1, 'finished': 0, 'ansible_job_id': 'j137356873998.105855', 'results_file': '/home/mattew/.ansible_async/j137356873998.105855', 'changed': True, 'item': {'image': 'quay.io/centos/centos:stream8', 'name': 'instance', 'pre_build_image': True}, 'ansible_loop_var': 'item'})

PLAY RECAP *****
localhost : ok=3 changed=2 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

INFO Running default > syntax
playbook: /home/mattew/ansible-roles/newrole/molecule/default/converge.yml
INFO Running default > create

PLAY [Create] *****

```

Рисунок 22 – Результат тестирования

Так как тестирование прошло успешно, можно запускать плейбук на управляемых узлах, рис 23.

```
-> trpp-ansible ansible-playbook ansible1.yml --ask-become-pass
BECOME password:

PLAY [Install and config Nginx] *****

TASK [Gathering Facts] *****
ok: [fed1]
ok: [fed2]

TASK [Install Nginx] *****
ok: [fed2]
ok: [fed1]

TASK [Delete default HTML files] *****
changed: [fed1]
changed: [fed2]

TASK [Replace config file] *****
ok: [fed1]
ok: [fed2]

TASK [Copy index file] *****
changed: [fed2]
ok: [fed1]

TASK [Generate dynamic HTML from template] *****
changed: [fed1]
ok: [fed2]

RUNNING HANDLER [Reload Nginx] *****
changed: [fed2]
changed: [fed1]

PLAY RECAP *****
fed1                : ok=7    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
fed2                : ok=7    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
-> trpp-ansible curl http://localhost
Web server is working!
```

Рисунок 23 – Результат выполнения плейбука и роли

Вывод

В ходе практической работы были получены навыки настройки вычислительной инфраструктуры с использованием системы управления конфигурациями Ansible. Созданы и настроены виртуальные машины, установлены необходимые компоненты, включая SSH и сам Ansible. Проведена базовая настройка inventory-файла, написан и успешно протестирован playbook для установки и конфигурации веб-сервера Nginx. Также реализована роль с использованием шаблонов и переменных, протестирована с помощью Molecule и применена к управляемым узлам. Работа позволила освоить как базовые, так и продвинутые возможности Ansible.