|  |
| --- |
|  |
| 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** |
| **«**Технологии разработки программных приложений**»**  **Тема: «**Системы конфигурационного управления**»** |

|  |  |
| --- | --- |
| Выполнил студент группы ИНБО-08-22 | Курышкин А. Д. |
| Старший преподаватель | Мельников Д. А.  . |
|  |  |

|  |  |  |
| --- | --- | --- |
| Практическая работа выполнена | «\_\_»\_\_\_\_\_\_\_ 2024г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_ 2024г. | *(подпись руководителя)* |

Москва 2024

**СОДЕРЖАНИЕ**

[1 ПОСТАНОВКА ЗАДАЧИ 3](#_Toc161783827)

[2 ХОД РАБОТЫ 4](#_Toc161783828)

[Подготовка инфраструктуры 4](#_Toc161783829)

[Результаты выполнения задания 9](#_Toc161783830)

[3 ВЫВОД 16](#_Toc161783831)

# 1 ПОСТАНОВКА ЗАДАЧИ

**Цель работы:**

Получить навыки настройки вычислительной инфраструктуры при помощи системы конфигурационного управления Ansible.

**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. 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:

13. postgresql-client

# **2 ХОД РАБОТЫ**

### Подготовка инфраструктуры

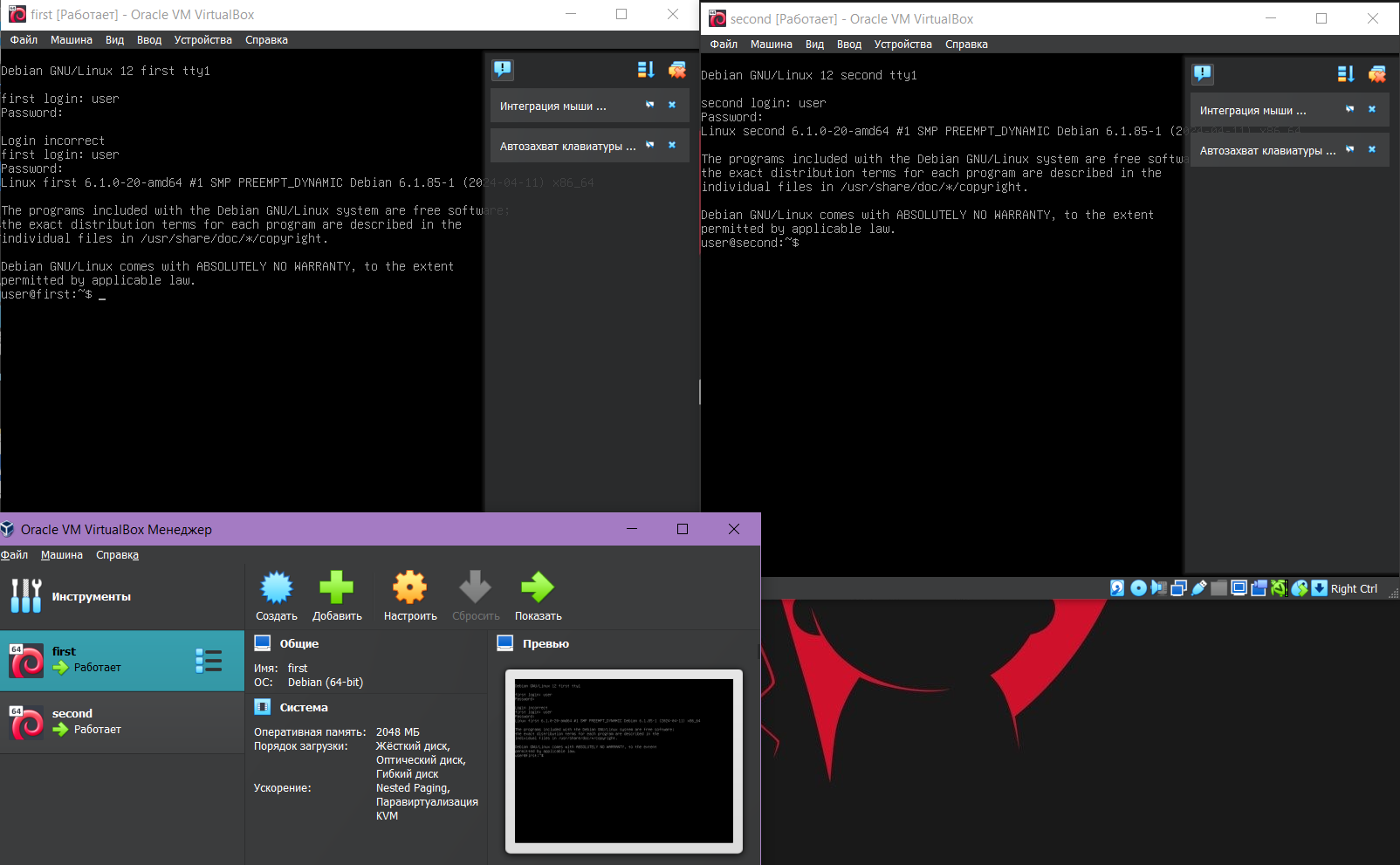
Прежде всего, необходимо создадим 2 виртуальные машины на Debian. Имя пользователя указано как user. Это важно для дальнейшей работы.

Рис. 1 – Две созданные виртуальные машины на Debian (управляемые)

После создания и установки ОС необходимо настроить доступ к сети. Для начала в VirtualBox измените тип сетевого подключения на сетевой мост. Это позволит вашей виртуальной машине находится в локальной сети на ровне с физическими устройствами. (Имя сетевого моста у каждого будет отличаться)

Получим адрес сети на хостовой машине с помощью команды ipconfig. Теперь можно приступить к настройке виртуальных машин.

Будем производить настройку при помощи файла /etc/network/interfaces.

* Параметр address назначается по ip, полученному для сети хостовой машины. К примеру, для ip 192.168.55.15 для физической машины, можно настроить ip 192.168.55.10 для первой управляемой машины и ip 192.168.55.20 для второй управляемой машины.
* Параметр mask обычно равен 255.255.255.0.
* Параметр gateway чаще всего является первым адресом в сети.

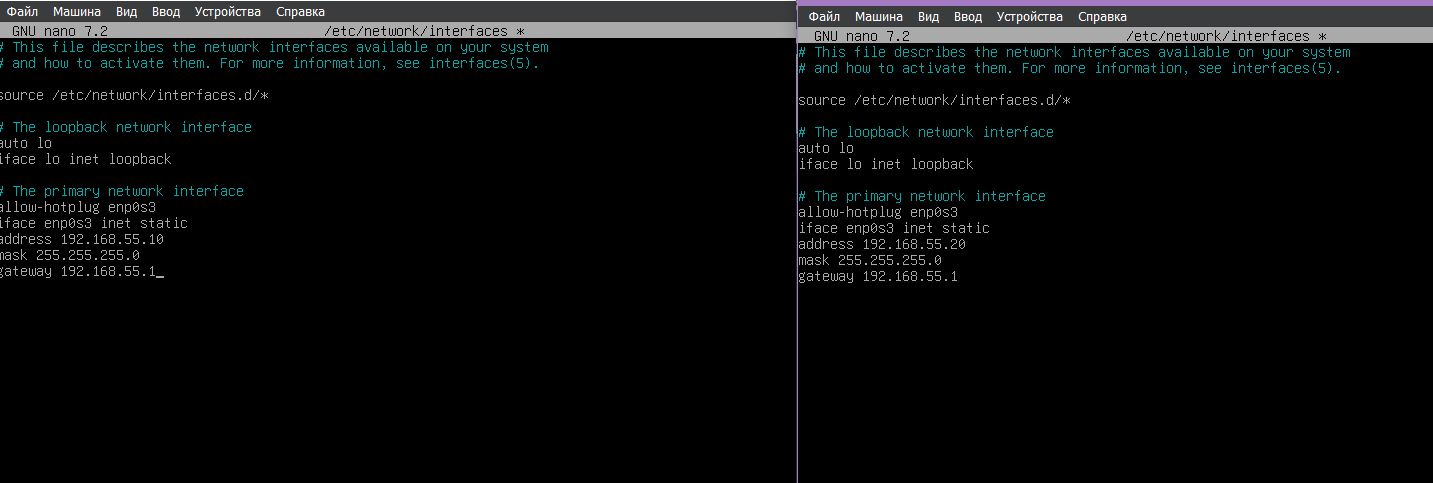


Рис. 2 – Настройка сети на управляемых машинах

В качестве хостовой машины используется еще одна виртуальная машина Debian



Рис. 3 – Созданная виртуальная машина на Debian (управляющая)

Произведем установку службы ssh для подключения к управляемым машинам

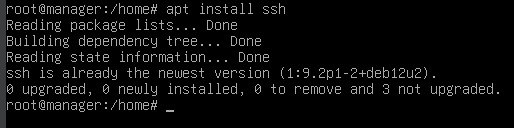


Рис. 4 – Процесс установки ssh на управляющую машину

Аналогично ssh установлен на управляемые машины.

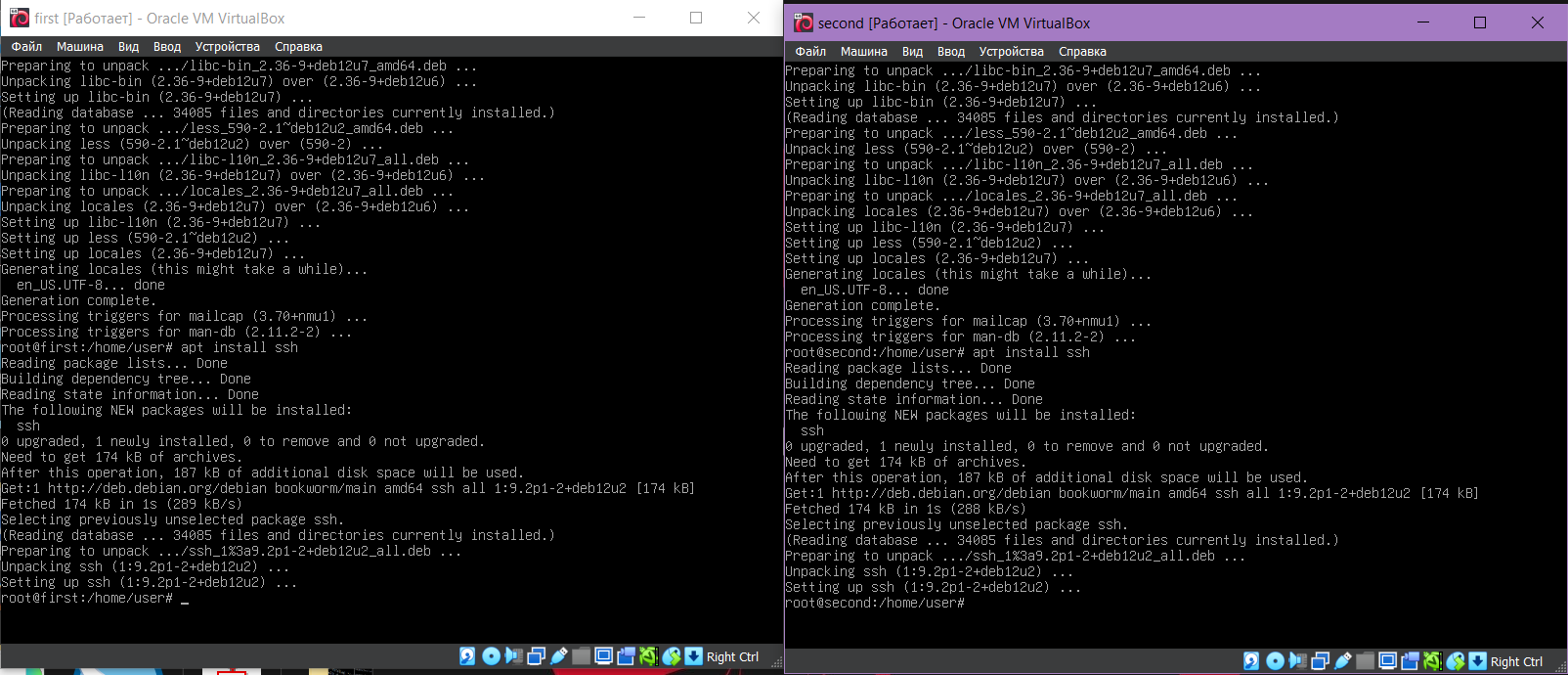


Рис. 5 – Процесс установки ssh на управляемые машины

После чего попробуем подключиться к управляемым машинам при помощи ssh.

ssh root@ip\_address

где ip\_address – то адрес управляемого узла в сети.

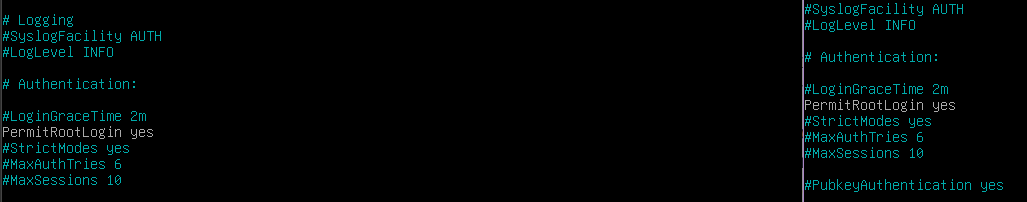
Если соединение удастся установить, после подключения система попросит ввести пароль пользователя.

Если нет, необходимо разрешить доступ на машину от пользователя root. Для этого на управляемых машинах необходимо открыть файл /etc/ssh/sshd\_config, найти строку PermitRootLogin, раскомментировать её и изменить значение на yes. Итоговая строка должна выглядеть так:

**PermitRootLogin yes**

После этого необходимо перезапустить службу ssh при помощи команды:

**service ssh restart**



Риc. 6 – Изменение конфигурационного файла для ssh



Рис. 7 – Перезапуск сервиса ssh для обновления конфигурации

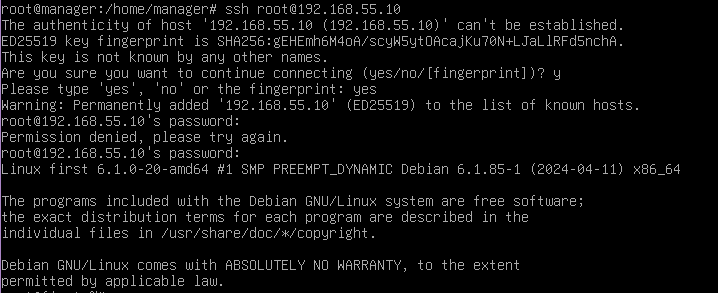


Рис. 8 – Успешное подключение с использованием ssh

Далее необходимо сгенерировать ключ при помощи команды ssh-keygen. Эта команда создаст ключ для возможности подключения к удаленному хосту без пароля.

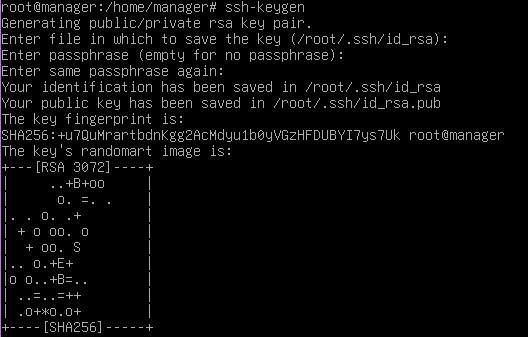


Рисунок 9 – Генерация ssh ключа

После этого необходимо воспользоваться командой ssh-copy-id для передачи ключа на обе виртуальные машины.

ssh-copy-id root@ip\_address

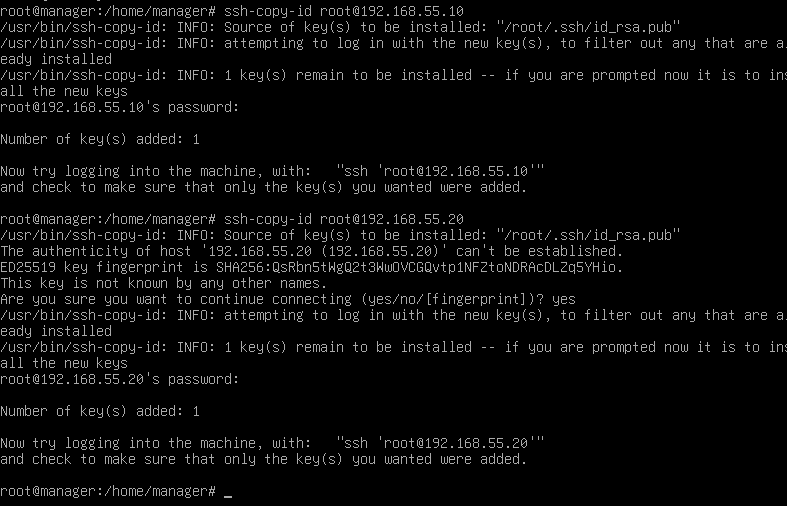


Рис. 10 – Копирование открытого ключа на управляемые машины

Теперь 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

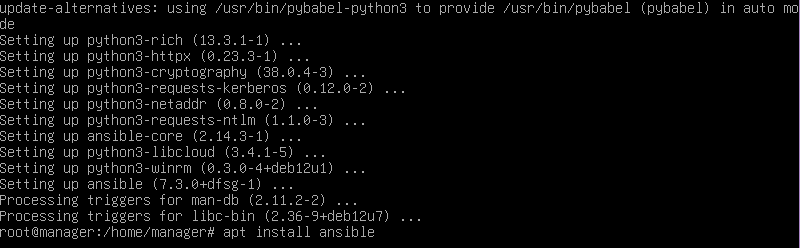


Рис. 11 – Установка Ansible с помощью пакетного менеджера apt

Наличие Ansible можно проверить командой ansible --version.

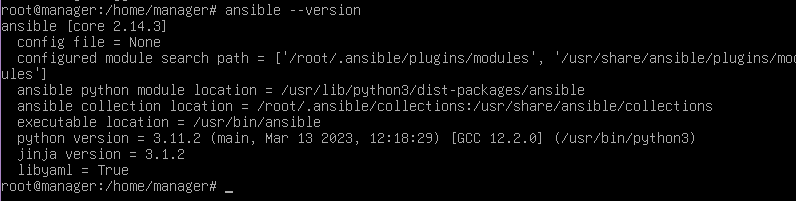


Рис. 12 – Просмотр версии Ansible на управляющей машине

Все действия выполняются 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.

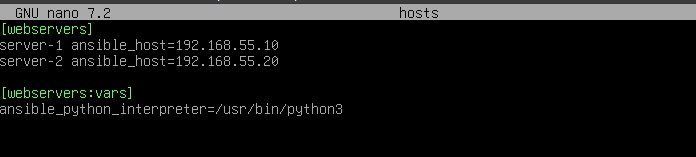


Рис. 13 – Содержимое файла hosts

webservers – это название группы хостов, управляемых при помощи Ansible.

webservers:var – список переменных для группы хостов webservers. В указанном примере задается переменная ansible\_python\_interpreter, которая указывает, какой интерпретатор python должен использоваться на управляемых узлах. Без указания этой переменной может возникнуть ошибка, если хост в качестве интерпретатора по умолчанию используется python второй версии, а не третьей.

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

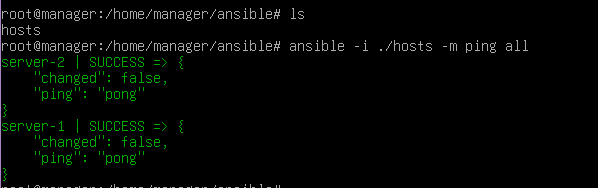


Рис. 14 – Результат вызова команды ping

Данная команда выполнит команду ping для всех хостов в inventory и выдаст результат выполнения.

Конкретную команду на хостах можно выполнить при помощи следующей конструкции:

ansible -i ./hosts -m command -a free all

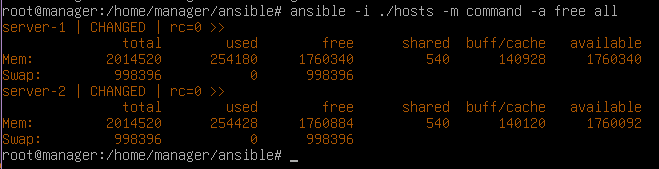


Рис. 15 – Результат вызова команды free

К примеру, данная команда покажет объемы свободной оперативной памяти на всех хостах.

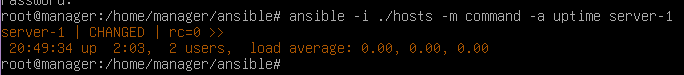


Рис. 16 – Результат вызова команды uptime на конкретной машине

**Результаты выполнения задания**

Механизм ролей Ansible 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 nginx при помощи команды

ansible-galaxy init nginx

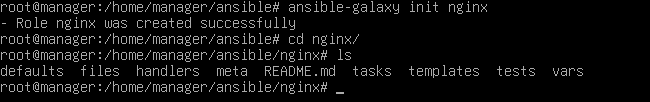


Рис. 17 – Результат вызова команды ansible-galaxy init

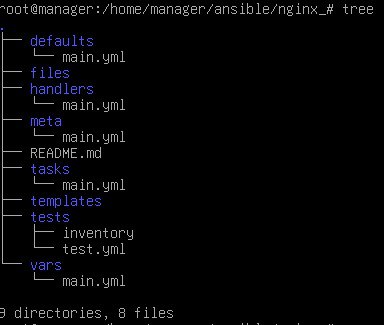


Рис. 18 – Начальное содержимое директории nginx

* 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 Ansilbe.
* Директория tasks содержит непосредственно сценарии для конфигураций.
* Директория templates, как можно догадаться, содержит шаблонные параметризованные файлы.
* Директория vars содержит файлы описания различных переменных.

Перейдём к созданию роли. Заполним соответствующие файлы данными из секций playbook’а и соответствующие директории ранее созданными файлами.

Это значит, что в директории tasks файл main.yml должен быть заполнен данными из секции tasks. В директории vars – из секции vars. В директории handlers – из секции handlers. Также файлы из директорий files и templates должны быть перемещены в директории files и templates в папке роли.

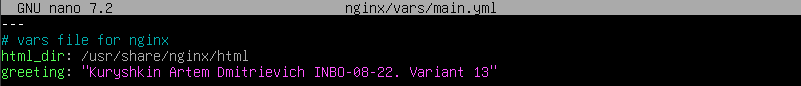
В разделе vars описываются переменные, которые затем можно использовать при запуске задач. Для использования переменных в задачах используется синтаксис шаблонизатора - {{ имя\_переменной }}.

Рис. 19 – Cодержимое файла main.yml

Ещё одним инструментом является использование шаблонных файлов. Для этого используется шаблонизатор Jinja2. Шаблонные файлы добавляются в директорию templates с дополнительных 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. 

Рис. 20 – содержимое файла hello.html.j2

Используем шаблонизированные файлы можно при помощи модуля template. В качестве параметров задается src – путь к шаблонному файлу, dest – пусть к файлу на управляемом узле, mode – режим доступа в 8-ричной системе счисления. Помимо этого, используется модуль register, позволяющий получить и проверить результат выполнения команды. Все результаты записываются в переменную result и при истинном значении параметра failed, означающего, что процесс переноса шаблона провалился, весь task становится проваленным благодаря конструкции failed\_when.

Также необходимо создать папку files, в которой будут храниться все статические файлы, к примеру файл index.html, которым будут заменены стандартные файлы nginx.

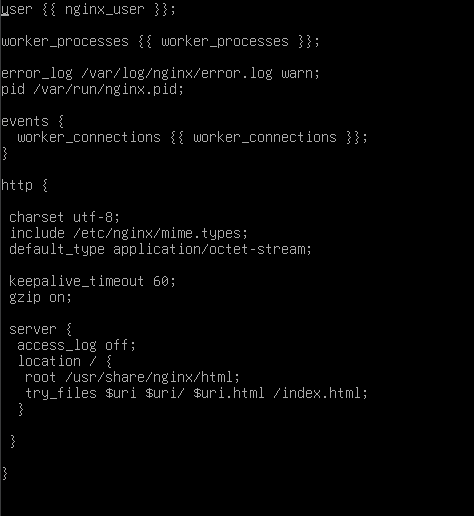


Рис. 21 – содержимое файла nginx.conf.j2

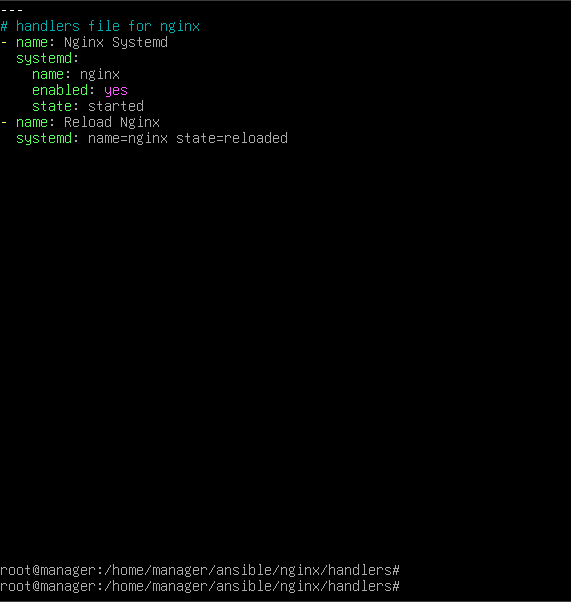


Рис. 22 – содержимое файла main.yml

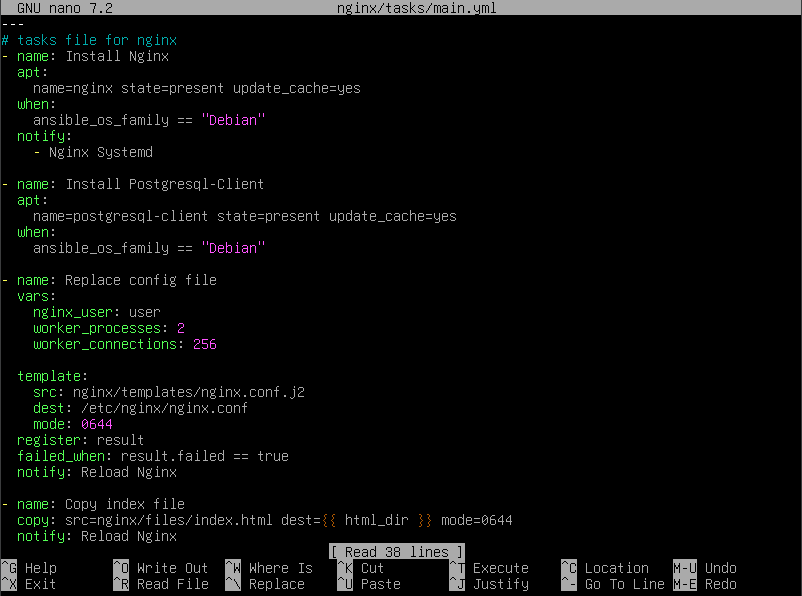


Рис. 23- Содержимое файла tasks/main.yml

Теперь напишем playbook, запускающий созданную роль. Для вызова составленной роли в playbook’е используется секция roles. Создадим файл nginx-role.yml в рабочей директории ansible и заполним его следующим образом.

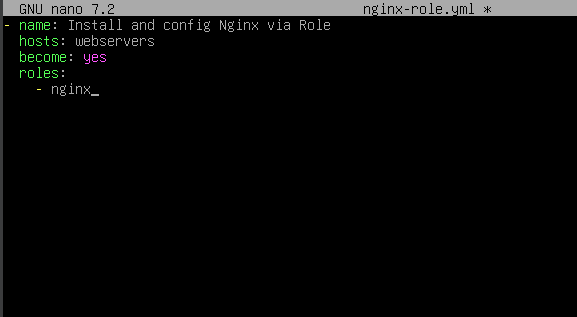


Рис. 24 – Содержимое файла nginx-role.yml

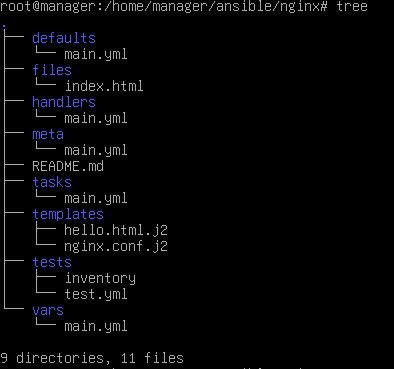


Рис. 25 - Конечное содержимое директории ansible

Проведем тестовый запуск playbook’а с помощью команды:

ansible-playbook -i hosts <имя файла playbook’а> --check

которая позволит проверить корректность написанного playbook’а без внесения изменений на целевые узлы.

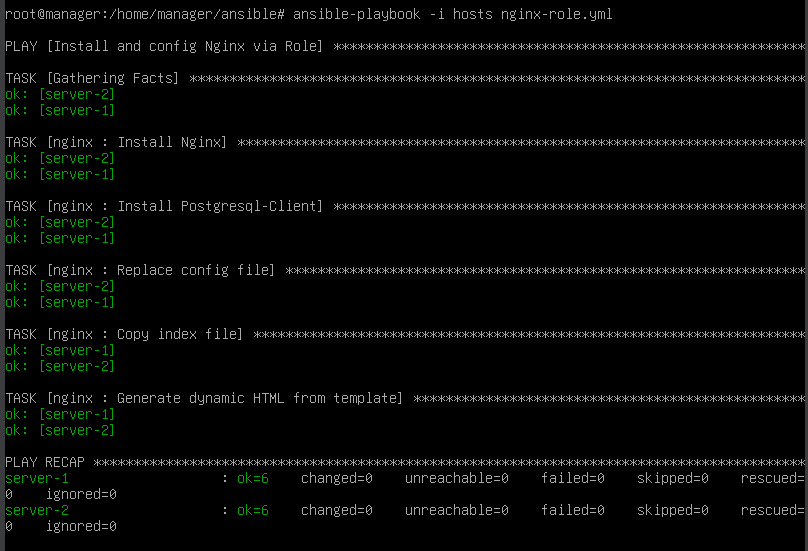


Рис. 26 – Результат тестового прогона

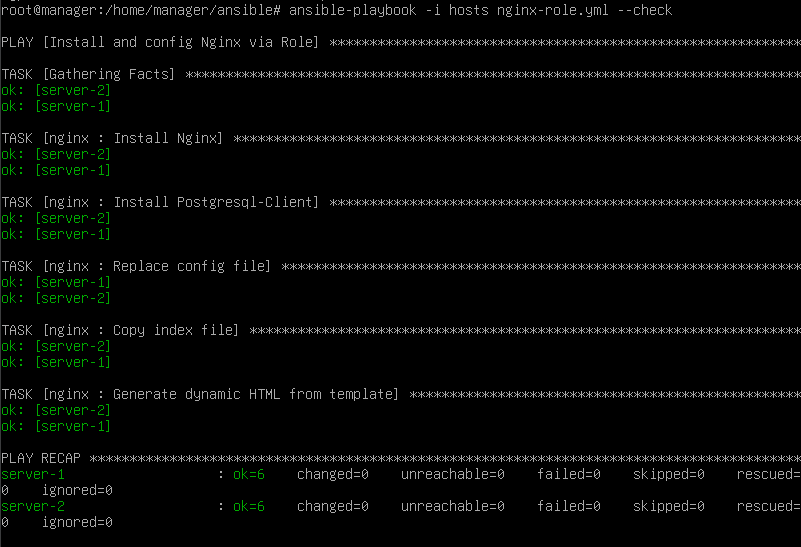
Теперь запустим роль на двух серверах и убедимся, что все работает корректно и соответствует ожидаемому результату.

Рис. 26 – Вывод команды ansible-playbook для описанной роли

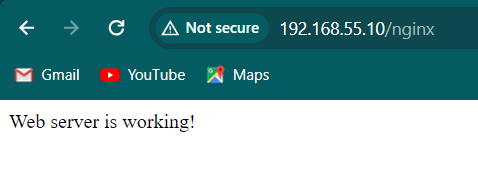


Рис. 27 – Результат по адресу <http://192.168.55.10>

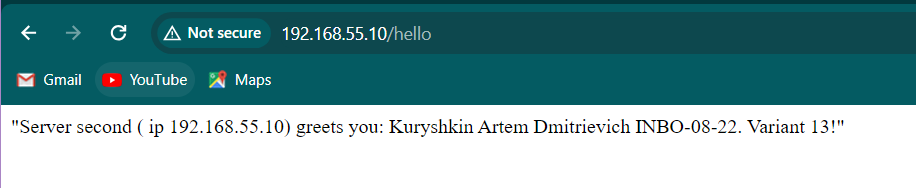


Рис. 28 – Результат по адресу <http://192.168.55.10/hello>

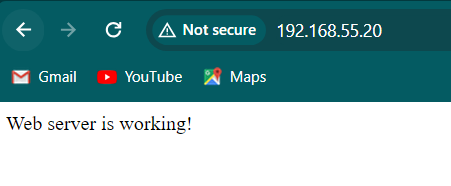


Рис. 29 – Результат по адресу <http://192.168.55.20>

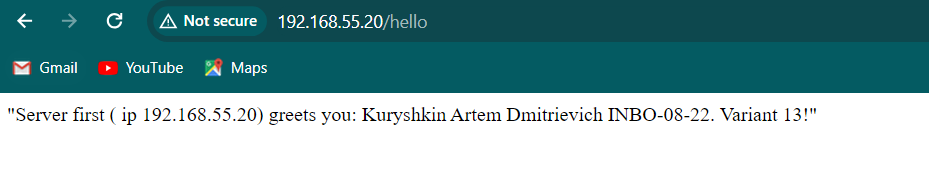


Рис. 30 – Результат по адресу http://192.168.55.20/hello

# 3 ВЫВОД

Роль для установки сервера nginx и переменной в шаблонном файле успешно разработана и протестирована.

Роль и playbook были успешно применены к серверам, и сервер nginx успешно запущен.

Переменная, содержащая ФИО, номер группы и номер варианта, была добавлена в шаблонный файл nginx и корректно отображается.

Также был добавлен task в playbook для установки пакета git в соответствии с вариантом.

Результаты тестирования и применения playbook'а подтверждают успешную настройку вычислительной инфраструктуры с помощью Ansible.