Введение в администрирование Linux: Ansible

Семинар #4 “Ansible: Расширение функционала”

Пантелеев Александр

Введение в администрирование Linux: Ansible

План занятия

- Ansible-lint
- Плагины
- Собственный фильтр
- Собственный модуль

Вопросы просьба задавать в конце подтемы.

Репозиторий с материалами лекции

ТЕНЗОР

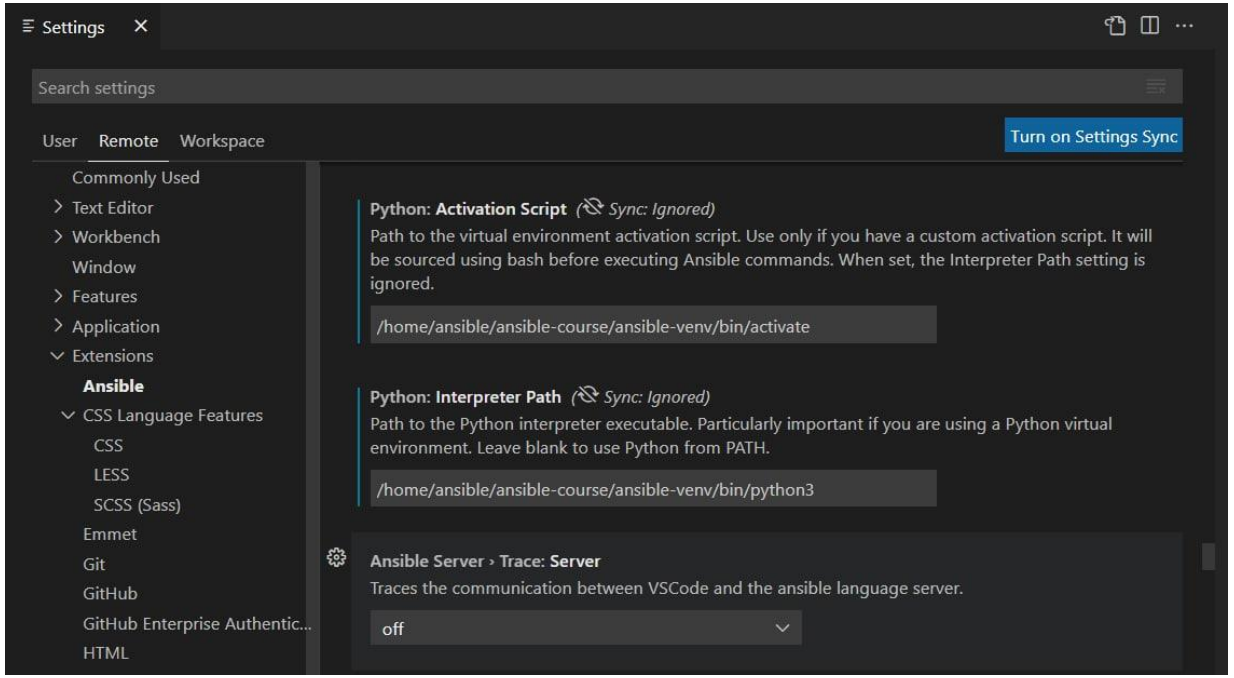
Ansible-lint

Ansible Lint – это инструмент командной строки, который анализирует код сценария на предмет потенциальных ошибок.

$ pip install ansible-lint yamllint

$ ansible-lint main.yaml

Настройки для расширения VSCode:
F1 → Preferences: Open Settings (UI) → Remote → Extensions → Ansible



Ссылка на документацию

Ansible расширение для VSCode

YAML расширение для VSCode

ТЕНЗОР

Плагины

Плагины в Ansible расширяют основной функционал Ansible. Плагины влияют на ход работы Ansible на управляющем сервере, модули – на удаленных серверах и возвращают результат на управляющий сервер.

Виды плагинов:

- Action
- Cache
- Callback
- Connection
- Lookup
- Shell
- Strategy
- Terminal
- Test
- Vars
- Filters
- Inventory

Ссылка на документацию

Ссылка на Github с исходным кодом

ТЕНЗОР

Callback-плагины

Ansible поддерживает два вида плагинов обратного вызова:

1. Плагины стандартного вывода (stdout plugins) преобразовывают информацию, выводимую в терминал
2. Другие плагины, выполняющие любые другие действия, не изменяющие вывода на экран (отправка в Slack, Telegram и т.д.)

Имя	Описание
actionable	Выводит только сообщения об изменениях и ошибках
debug	Выводит содержимое stderr и stdout в удобочитаемом виде
default	Отображает вывод по умолчанию
dense	Затирает старый вывод вместо прокрутки
json	Выводит информацию в формате JSON
minimal	Выводит результаты выполнения задач с минимальным форматированием
oneline	Минимальное форматирование в одну строку
selective	Отображает вывод только задач, отмеченных тегом print_action
skippy	Подавляет вывод для пропущенных хостов

Посмотреть список через команду
ansible-doc -t callback -l
Почитать подробнее через команду
ansible-doc -t callback <plugin name>

ТЕНЗОР

Strategy-плагины

linear: Задачи запускаются по очереди. Ansible ждет, пока задача выполнится на всех серверах перед тем, как приступить к следующей.
free: Ansible не ждет, пока задача выполнится на всех серверах перед тем, как приступить к следующей.

```yaml
---
- name: linear
  hosts: nodes
  connection: local
  gather_facts: false
  tasks:
    - name: first task
      shell: sleep "{{ sleep_seconds }}"

    - name: second task
      shell: sleep "{{ sleep_seconds }}"

    - name: third task
      shell: sleep "{{ sleep_seconds }}"

```

```yaml
---
- name: free
  hosts: nodes
  connection: local
  strategy: free
  gather_facts: false
  tasks:
    - name: first task
      shell: sleep "{{ sleep_seconds }}"

    - name: second task
      shell: sleep "{{ sleep_seconds }}"

    - name: third task
      shell: sleep "{{ sleep_seconds }}"

```

ansible-playbook -i hosts 02b_ansible_free_strategy.yaml

ТЕНЗОР

Strategy-плагины

debug: Ansible содержит в себе интерактивный отладчик, который может помочь отлавливать ошибки во время выполнения сценария.

```
p task
p task_vars['var3']
task_vars['var3'] = 'value3'
p task_vars['var3']
update_task
redo
```

```yaml
- name: ansible-debugger
  hosts: nodes
  strategy: debug
  gather_facts: true
  vars:
    var1: value1
    var2: value2
  tasks:
    - name: task without variable
      debug:
        msg: "{{ var3 }}"

```

[Ссылка на документацию]

ТЕНЗОР

Lookup-плагины

Используются для динамического получения информации из внешних источников(файлы, системы управления секретами, переменные окружения)

Могут использоваться для генерации циклов.

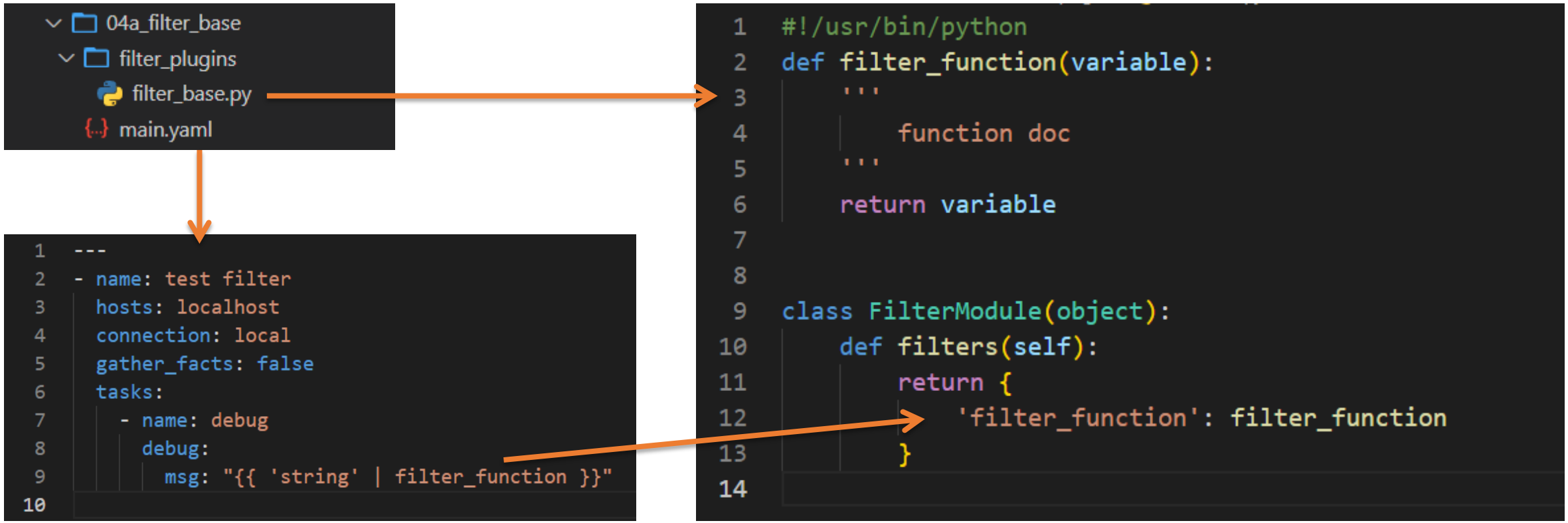
```yaml
---
- name: lookup plugins
  hosts: nodes
  gather_facts: false
  vars:
    users:
      alice:
        name: Alice Appleworth
        telephone: 123-456-7890
      bob:
        name: Bob Bananarama
        telephone: 987-654-3210
  tasks:
    - name: file lookup
      debug:
        msg: "file contents: {{ lookup('file', './04b_file_for_lookup.txt').split('\n') }}"
    - name: env lookup
      debug:
        msg: "env contents: {{ lookup('env','example_env_var') }}"
    - name: output dict lookup result
      debug:
        msg: "{{ lookup('dict', users) }}"
    - name: Print phone records
      debug:
        msg: "User {{ item.key }} is {{ item.value.name }} ({{ item.value.telephone }})"
      loop: "{{ lookup('dict', users) }}"
    - name: set_fact when alice in key
      debug:
        msg: "User alice exists"
      loop: "{{ lookup('dict', users) }}"
      when: "'alice' in item.key"

```

ansible-doc -t lookup -l

[Ссылка на документацию]

ТЕНЗОР

Практика: Пишем фильтр



[Ссылка на документацию]

ТЕНЗОР

Домашнее задание

- Написать собственный фильтр для преобразования mac адреса: 2056876429 -> 20:56:87:64:29.
- Задание со звездочкой - использование regexp.
- Добавить исключения: Проверка, что в фильтр передается строка. Проверка, что в строке четное количество символов. Проверка допустимых символов (цифры от 0 до 9, буквы от A до F)

ТЕНЗОР

Практика: Пишем модуль

```yaml
- name: test module
  hosts: localhost
  connection: local
  gather_facts: false
  tasks:
    - name: Example module
      example_sh:
        string1: value1
      register: bash_result

    - name: print result
      debug:
        msg: "{{ bash_result }}"

    - name: Example module
      example_py:
        string1: value1
      register: py_result

    - name: print result
      debug:
        msg: "{{ py_result }}"
```

```bash
#!/bin/bash
# WANT_JSON

string1=$(cat $1 | grep -Po '(?<="string1": ")(.*?)(?=")')

result_string="$string1"
# sleep 300

echo "{\"result_str\": \"$result_string\", \"msg\": \"Success\"}"
```

```python
def main():
    # Задаем аргументы модуля
    module_args = dict(
        string1=dict(required=True, type='str')
    )
    # Создаем объект - модуль
    module = AnsibleModule(
        argument_spec=module_args,
        supports_check_mode=False
    )
    # Получаем из модуля аргументы
    string1 = module.params["string1"]
    # Вызываем нашу функцию
    lc_return = example_function(string1)
    # Если задача зафейлилась
    if lc_return[0]:...
    # Если задача успешно завершилась
    else:...
```

```python
def example_function(str1):
    failed = False
    msg = "Success"
    rc = 0
    # Формируем конечную строку
    try:
        result = str1
    except TypeError as e:
        failed = True
        result = ""
        msg = "TypeError. str1 is not a string"
        rc = 1
    return(failed, result, rc, msg)
```

[Ссылка на документацию]

ТЕНЗОР

Домашнее задание

Написать на Python или Bash свой модуль, который вернет код состояния HTTP-сервера.
Дополнительный балл, если напишете оба модуля.
Основа для написания модулей в репозитории по ссылке.

Итоговое домашнее задание

К домашнему заданию из третьего занятия добавить две новые роли по настройке PHP и Wordpress.
Внести необходимые изменения в роли Nginx и Mariadb.
В репозитории с учебными материалами для 4 занятия в папке HW находятся:

- Все необходимые .repo-файлы и список пакетов для установки PHP 7.4
- Конфигурационные файлы для Nginx, PHP и Wordpress
- Примерный список задач, необходимых для окончательной настройки Wordpress.

ТЕНЗОР

Познакомились с плагинами.
Написали собственный фильтр, модуль.

Вопросы?